

Cloud Object Storage

Developer Guide

Product Documentation



Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

Developer Guide

Creating Request

Overview

Bucket

Bucket Overview

Creating a Bucket

Deleting a Bucket

Object

Object Overview

Storage Class

Storage Class Overview

DEEP ARCHIVE Overview

Overview - INTELLIGENT TIERING

Uploading Object

Simple Upload

Multipart Upload

Upload via Pre-Signed URL

Downloading Object

Simple Download

Download via Pre-Signed URL

Listing Object Keys

Copying Objects

Simple Copy

Multipart Copy

Deleting Objects

Deleting a Single Object

Deleting Multiple Objects

Data Management

Lifecycle Management

Lifecycle Overview

Lifecycle Configuration Elements

Configuring Lifecycle

Hosting a Static Website

Inventory Overview

Bucket Tag Overview

Object Tag Overview

Event Notifications

Data Extraction

 SELECT Overview

 SELECT Command

 SQL Functions

 Reserved Words

 Data Types

 Operators

Log Management

 Logging Overview

 Restrictions on Log Management

 Using COS to Store Tencent Cloud Product Logs

Data Disaster Recovery

 Versioning

 Overview

 Delete Markers

 Using Versioning

 Versioning Configuration

 Cross-Bucket Replication

 Overview

 Actions

 Configuration

 MAZ Feature Overview

Data Security

 Server-side Encryption Overview

 Bucket Encryption Overview

 Server-side Encryption Overview

 Object Lock Overview

Cloud Access Management

 Access Permission Configuration Description

 Access Control Overview

 Basic Concepts of Access Control

 COS Authorization and Identity Verification Process

 Notes on Principle of Least Privilege

 Access Policy Evaluation Process

 Access Control Methods

 Bucket Policy

- User Policy

- ACL

- Tag-based project resource management

- Authorization Methods

- Access Policy Language

- Overview

- Conditions

- Request Methods

- Accessing COS Using a Permanent Key

- Accessing COS Using a Temporary Key

- Accessing COS Using a Pre-Signed URL

- Accessing COS Anonymously

- Using CDN to Accelerate Access

- CDN Acceleration Overview

- Setting CDN Acceleration

- Single-Connection Bandwidth Limit

- Batch Operation

- Managing Batch Operation Jobs

- Overview

- Performing Batch Operation

- Batch Copying Objects

- Batch Restoring Archived Objects

- Global Acceleration

- Overview

- Private Network Global Acceleration

- Data Workflow

- Overview

- Monitoring and Alarms

Developer Guide

Creating Request

Overview

Last updated : 2024-03-25 15:28:24

Concept

Tencent Cloud COS is a web-based storage service accessed using the HTTP/HTTPS protocol. You can use [RESTful APIs](#) or [COS SDKs](#) to access COS.

Your COS access request must first pass the COS identification and authentication before COS starts to operate the resources. Therefore, depending on whether the identity is identifiable, COS access requests are divided into two types: anonymous request and signed request.

Anonymous request: If a request does not include `Authorization` or related parameters, or the user identity cannot be identified based on the related characters, the request will be treated as an anonymous request for authentication.

Signed request: A signed request must contain the `Authorization` field in the HTTP header or the request packet. The content of the field is generated based on Tencent Cloud security credentials (SecretID and SecretKey) and some characteristic values of the request via an encryption algorithm.

To access COS using COS SDKs, you only need to configure your security credentials before initiating the request. To access COS using the REST API, calculate the request signature according to [Request Signature](#).

Obtaining Security Credentials

Cloud Access Management (CAM) provides features and services related to accounts and credentials for COS, to help customers manage the permissions to access resources under their Tencent Cloud accounts in a secure way. You can use CAM to create, manage and terminate users (or user groups), and manage other users' permissions to use Tencent Cloud resources through identity management and policy management.

Security credentials of the root account

After logging in to the root account, you can manage and obtain the security credentials (SecretID and SecretKey) of your root account on the [Cloud API Key](#) page of CAM. The following is a key pair example:

36-character access key ID (SecretID): AKIDHZRLB9lbhdp7Y7gyQq6BOK1997xxxxxx

32-character access Key (SecretKey): LYaWluQmCSZ5ZMniUM6hiaLxHnxxxxxx

The access key can be used to identify the uniqueness of an account. After the signature is generated using the key and the request is sent, Tencent Cloud will identify the identity of the request initiator, and then perform verification and authentication for the identity, resources, operations, and conditions to determine whether to allow the operation.

Note:

The key of the root account has all the operation permissions for all resources under the root account. Disclosure of the key may cause loss of your cloud assets, so it is strongly recommended that you create sub-accounts and assign corresponding permissions for them, and then use the keys of sub-accounts to create requests for resource access and management.

Security credentials of sub-accounts

To manage users and cloud resources under your account in multiple dimensions, you can create multiple sub-accounts under your primary account to implement user-specific permission management. For more information on how to create a sub-account, see [Sub-users](#) in CAM.

Before using a sub-account to initiate an API request, you need to create a security credential for the sub-account, and then the sub-account will get a unique key pair, which can facilitate the identification of the identity. You can create user policies for different sub-accounts to control their access permissions to resources. You can also create user groups and associate one access policy to a user group to facilitate the central management of user grouping and resources.

Note:

With the corresponding permissions assigned, a sub-account can create or modify resources. The resources still belong to the primary account, and the resource cost will be deducted from the root account.

Temporary security credentials

In addition to using security credentials of the root account or sub-accounts to access resources, you can create roles and use the temporary security credentials of the roles to manage your Tencent Cloud resources. For more information on the role concept and how to use roles, see [Role Overview](#).

As a virtual identity, a role does not have a permanent key. Tencent Cloud CAM provides a set of STS APIs used to generate temporary security credentials.

For more information on how to use the APIs and relevant examples, see [Using Roles](#). See [CreateRole](#) to learn about how to generate temporary security credentials. Temporary security credentials contain only **limited policies** (operations, resources, and conditions), and are valid for a **limited period** (start and end time), so the generated temporary security credentials can be distributed or used directly.

You can call the API for generating temporary security credentials and get a temporary key pair (tmpSecretId/tmpSecretKey) and a security token (sessionToken), which form the security credential that can be used to access COS. The following is an example of a temporary security credential:

41-character security token (SecurityToken): 5e776c4216ff4d31a7c74fe194a978a3ff2xxxxxx

36-character temporary access key ID (SecretID): AKIDcAZnqgar9ByWq6m7ucIn8LNEuYxxxxxx

32-character temporary access key (SecretKey): VpxrX0IMCpHXWL0Wr3KQNCqJixxxxxxx

This API also returns the validity period of the temporary security credential via the `expiration` field, which means that this set of security credentials can only be used to initiate requests during this period.

Tencent Cloud COS provides a simple server-side SDK that can be used to generate temporary keys. You can visit [COS STS SDK](#) to obtain the SDK. To initiate the request using the REST API after getting the temporary security credential, you need to specify the value for the `x-cos-security-token` field in the HTTP header or the form-data of the POST request packet to identify the security token used by the request, and then use the temporary access key pair to generate the request signature. For more information on how to initiate requests using the COS SDK, see the relevant sections in each SDK documentation.

Access Endpoint Domain Names

RESTful APIs

The [Region and Access Endpoints](#) document provides a list of domain names that can be used to initiate access requests via RESTful APIs.

We recommend you use virtual hosting domain names to access COS buckets. When you initiate an HTTP request, the bucket to be accessed will be specified through the `Host` header, for example, `<BucketName-APPID>.cos.<Region>.myqcloud.com`. Using virtual hosting domain names implements the same feature as the root directory of a virtual server. Virtual hosting domain names can be used to host files such as `favicon.ico`, `robots.txt`, and `crossdomain.xml`, which are the content that many applications will retrieve from the root directory of the virtual server by default when identifying a hosted website.

You can also use a path request to access a bucket, for example, `cos.`

`<region>.myqcloud.com/<BucketName-APPID>/`. The request `Host` and the signature must use `cos.<region>.myqcloud.com`. COS SDKs do not support this access method by default.

Domain names of static websites

If you enable the static website feature for a bucket, a virtual hosting domain name will be assigned for you to use relevant features. Unlike RESTful APIs, the domain name of a static website supports only a few operations, such as GET/HEAD/OPTIONS Object, in addition to specific index pages, error pages and redirection configurations. Uploading or configuring resources is not supported.

The format of a domain name of a static website is `<BucketName-APPID>.cos-website.`

`<Region>.myqcloud.com`. You can also log in to the console and go to the bucket's **Basic Configuration** > **Static Website Configuration** to get the domain name.

Private network access

Tencent Cloud COS adopts intelligent resolution for COS endpoints. In this way, the optimal linkage can be provided for you to access COS with different ISPs.

If you have deployed a CVM within Tencent Cloud for accessing COS over a private network, you must first ensure that the CVM resides in the same region as the COS bucket, then use the `nslookup` command on the CVM to resolve the COS endpoint. If a private IP is returned, access between the CVM and COS is over a private network; otherwise, it is over a public network.

If your CVM resides in a different region from the COS bucket, but it is still in one of COS regions, you can use the COS private network global acceleration domain name to access files and achieve cross-region access between the CVM and COS.

How to determine whether access is via a private network

Tencent Cloud products within the same region can access each other over a private network, incurring no traffic fees. Therefore, we recommend choosing the same region when you purchase different Tencent Cloud products to save on costs.

Note:

The private networks of Public Cloud regions do not interconnect with those of Finance Cloud regions.

The following shows how to determine access over a private network:

For example, when a CVM accesses COS, to determine whether a private network is used for access, use the `nslookup` command on the CVM to resolve the COS endpoint. If a private IP is returned, access between the CVM and COS is over a private network; otherwise, it is over a public network.

Note:

Generally, a private IP takes the form of `10.*.*.*` or `100.*.*.*`, and a VPC IP takes the form of `169.254.*.*`. These two types of IPs belong to private networks.

Assume that `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com` is the address of the destination bucket. After running the `nslookup` command, you can view the information as shown in the figure below.

```
nslookup examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com
Server:      10.138.224.65
Address:     10.138.224.65 #53
Name:        examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com
Address:     10.148.214.13
Name:        examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com
Address:     10.148.214.14
```

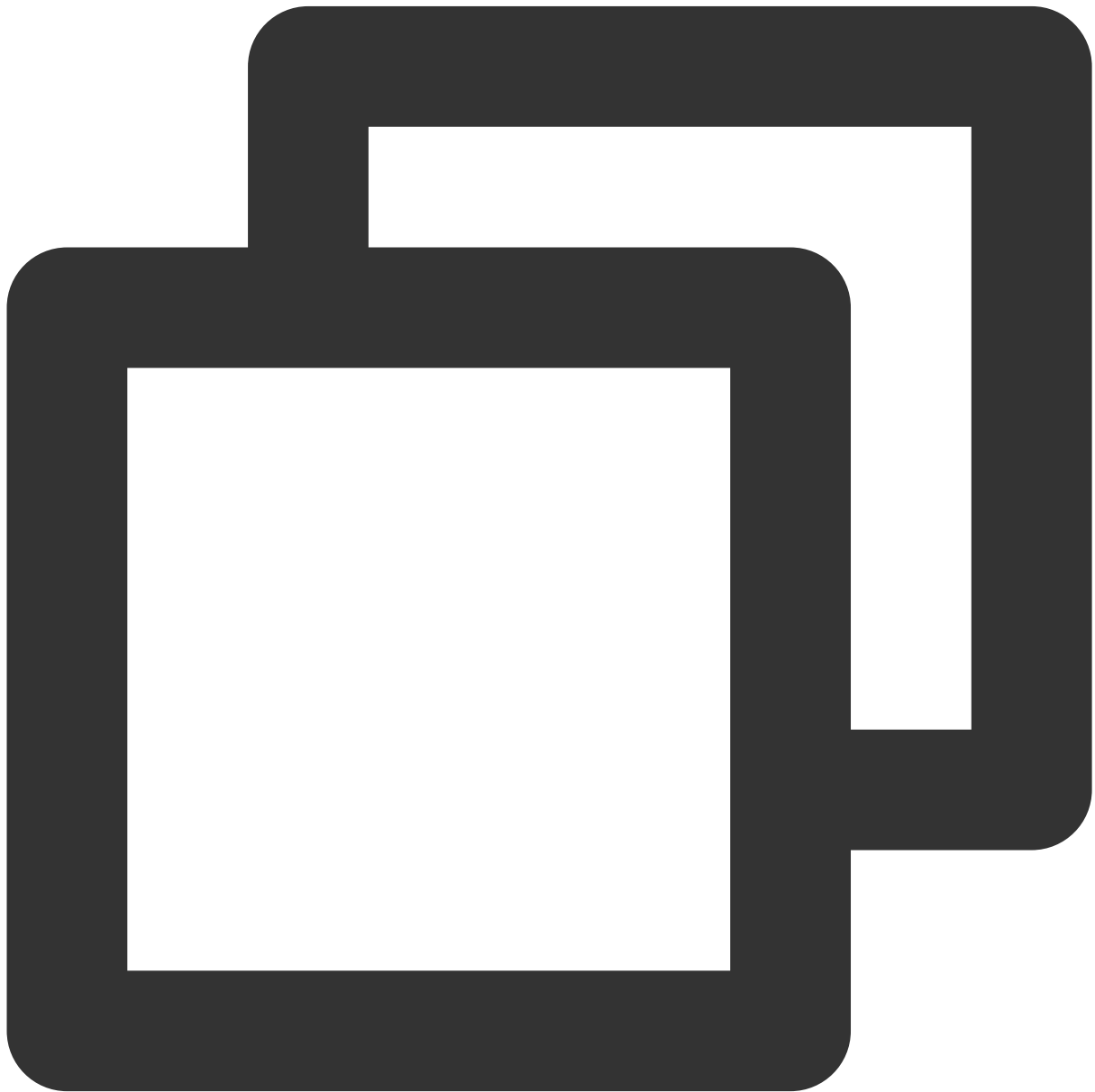
In the command output, the `10.148.214.13` and `10.148.214.14` IPs indicate that the access to COS is over a private network.

Testing connectivity

Basic connectivity test

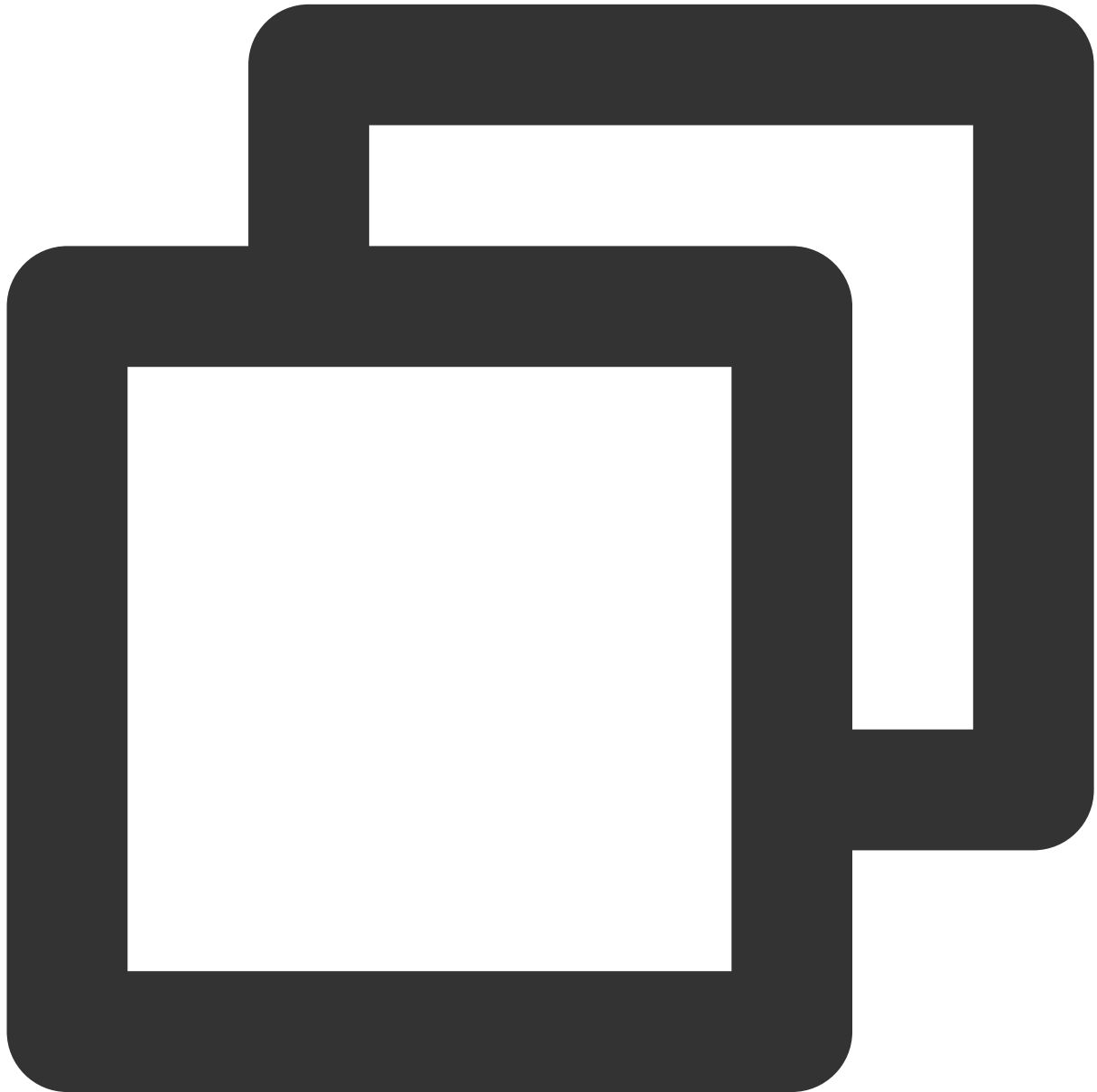
COS uses the HTTP protocol to provide services. You can use the most basic tool `telnet` to test the connectivity to port 80 of the COS access domain.

The following is an example of access through the public network:



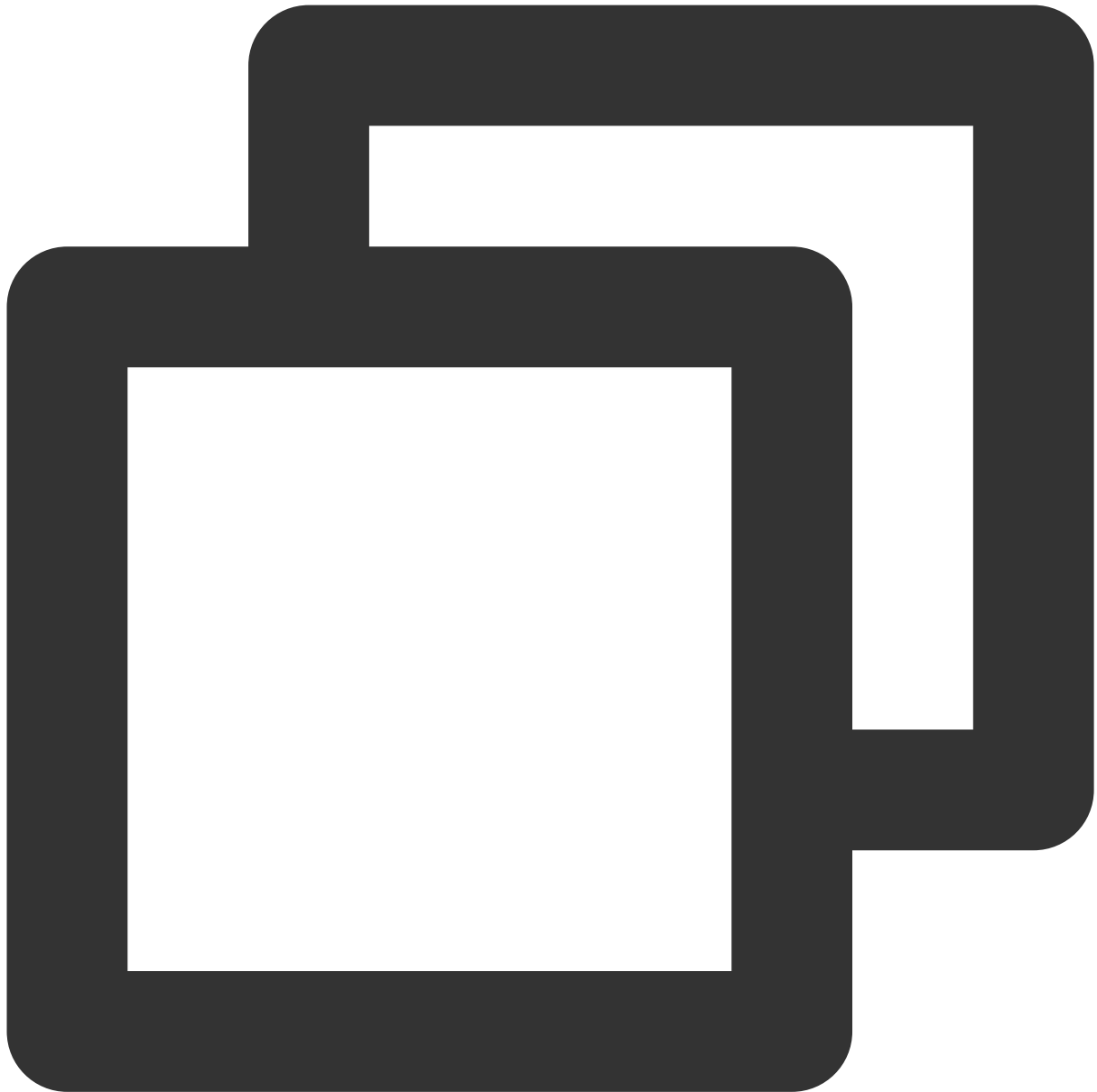
```
telnet examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com 80
Trying 14.119.113.22...
Connected to gz.file.myqcloud.com.
Escape character is '^]'.
```

The following is an example of access through Tencent Cloud CVMs (classic network) within the same region:



```
telnet examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com 80
Trying 10.148.214.14...
Connected to 10.148.214.14.
Escape character is '^]'.
```

The following is an example of access through Tencent Cloud CVMs (VPC) within the same region:



```
telnet examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com 80
Trying 169.254.0.47....
Connected to 169.254.0.47.
Escape character is '^]'.
```

Regardless of the access environment, if the command returns the `Escape character is '^]'.` field, it indicates that the connection is successful.

Test via the internet

The access to COS over the internet involves the ISP network, which may prohibit you from testing connectivity using tools such as ICMP `ping` or `tracert`. Therefore, we recommend you use TCP tools to test connectivity.

Note:

The access via the Internet may involve multiple network environments. If the access is not smooth, check your local network linkage, or contact the local ISP.

If your ISP allows you to use the ICMP protocol, you can use the `ping`, `tracert` or `mtr` tools to check your linkage. Otherwise, you can use the `psping` (Windows environment; download at the Microsoft official website) or such tools as `tcpping` (cross-platform software) to test the latency.

Test via a private network

If you access the COS over the Tencent Cloud VPC in the same region, you may be unable to test connectivity using tools such as ICMP `ping` or `tracert`. We recommend that you use the `telnet` command in the basic connectivity test to perform the testing.

You can also use tools such as `psping` or `tcpping` to test the latency to port 80 of the access domain. Before the test, make sure that the access domain name has been correctly resolved to the private network address using the `nslookup` command.

Bucket

Bucket Overview

Last updated : 2024-03-25 15:28:24

Overview

A bucket is the carrier of objects, which can be understood as the "container" for storing objects, and this "container" has no upper limit of capacity. Objects are stored in buckets in a flat structure with no concept of folders and directories. You can choose to store objects in one or multiple buckets.

Note:

A bucket can contain any number of objects, but one root account can create only up to 200 buckets.

Bucket Naming Conventions

A bucket name consists of BucketName and APPID connected by a hyphen (-). For example, in the bucket name `examplebucket-1250000000`, `examplebucket` is the user-defined string, and `1250000000` is the system-generated numeric string (APPID). In API and SDK samples, the naming format of a bucket is `<BucketName-APPID>`.

BucketName: a custom string of characters in the following conventions:

Only lowercase letters (a-z), digits (0-9), and hyphens (-) are allowed.

The number of characters a bucket name is allowed to contain is limited by the length of the [region abbreviation](#) and **APPID**. The combined domain name can be up to 60 characters. For example, the domain name

`123456789012345678901-1250000000.cos.ap-beijing.myqcloud.com` contains 60 characters.

A bucket name cannot start or end with "-".

APPID: the account you get after you successfully register in Tencent Cloud. It is automatically assigned by the system as a unique permanent ID, which can be viewed in [Account Information](#). When creating a bucket in the console, you don't need to enter it; however, when using tools, APIs, and SDKs, you need to specify it.

The following are examples of valid bucket names:

`examplebucket-1-1250000000`

`mybucket123-1250000000`

`1-newproject-1250000000`

Bucket Region

Region is where the COS IDC is located. COS allows users to create buckets in different regions. You can select the region closest to the location where you deploy your business for the buckets so as to reduce latency and cost, and meet the compliance requirements.

For example, if your business is distributed in South China, creating buckets in the Guangzhou region can accelerate object uploads and downloads. For more information on regions, see [Regions and Access Endpoints](#).

Note:

A region must be specified when a bucket is created and cannot be modified once specified. All objects in the bucket are stored in the IDC in the region. You cannot set regions for objects.

Types of Permission

A bucket provides two types of permissions by default: public and user.

Note:

If the bucket permission is private read/write or a specified account is granted the permission, an object request needs to carry a signature for identity verification. For more information on signature, see [Request Signature](#).

If the bucket permission is public read/private write or public read/write, an object request doesn't need to carry a signature, and anonymous users can directly access the object at the URL. However, your data may be leaked.

Therefore, proceed with caution.

Public permissions

Public permissions include "Private Read/Write", "Public Read/Private Write", and "Public Read/Write". You can modify bucket access permissions in **Permission Management** of the bucket in the COS console. For more information, see [Basic Concepts of Access Control](#).

Private Read/Write

Only the creator of the bucket and authorized accounts have Read/Write permission on the objects in the bucket. The default access permission of a bucket is Private Read/Write, which is recommended.

Public Read/Private Write

Anyone (including anonymous visitors) has Read permission on the objects in the bucket, but only the bucket creator and authorized accounts have Write permission on them.

Public Read/Write

Anyone (including anonymous visitors) has Read/Write permission on the objects in the bucket, which is not recommended.

User permissions

A root account has all the permissions (full access) for buckets by default. In addition, you can add sub-accounts that are granted permissions to read/write data and permissions, and even full access.

Bucket Operations

You can manage buckets and configure attributes of buckets in various methods such as the Tencent Cloud console, tools, APIs, and SDKs. For example, you can set a bucket for hosting a static website or set access permission on a bucket. The following documents describe how to configure some features. For more information on bucket configuration, see [Bucket Overview](#).

[Creating a Bucket](#)

[Setting Up a Static Website](#)

[Setting Access Permission](#)

[Setting Hotlink Protection](#)

Notes

COS stores objects using a flat structure instead of folders. For more information, see “Folders and Directories” in [Object Overview](#).

Each root account (i.e., the same APPID) can create up to 200 buckets in total in all regions. There is no limit on the number of objects in a bucket.

In Tencent Cloud COS, the bucket name under one APPID must be unique.

Once a bucket is created, it cannot be renamed. To rename a bucket, you need to delete it and create another one with the desired name.

When creating a bucket, make sure to select the desired region, as the region cannot be changed once specified.

Creating a Bucket

Last updated : 2024-03-25 15:28:24

To store objects in COS, you first need to create a bucket using the console, tools, APIs, or SDKs.

After creating a bucket, you can upload objects to it and configure other features for it, such as [setting up a static website](#), [setting bucket tags](#), and [setting bucket encryption](#). For more configuration instructions, see [Console Overview](#).

Restrictions

1. One root account can create up to 200 buckets.
2. Once a bucket is created successfully, its name and region cannot be modified.
3. Bucket names under the same root account must be unique and cannot be changed.
4. A bucket name can contain letters, digits, and hyphens (-). The number of characters a bucket name is allowed to contain is limited by the length of the [region abbreviation](#) and **APPID**. The combined domain name can be up to 60 characters. For example, the domain name `123456789012345678901-1250000000.cos.ap-beijing.myqcloud.com` contains 60 characters.
5. A bucket name cannot start or end with "-".

Directions

Using COS console

Create a bucket in the COS console. For more information, see [Creating a Bucket](#).

Using tools

Use tools such as COSBrowser and COSCMD to create a bucket. For more information, see [Tool Overview](#).

Using REST APIs

Use the REST API to initiate a bucket creating request. For more information, see [PUT Bucket](#).

Using SDKs

Directly call the bucket creating method in the SDK. For more information, see the SDK documentation for the corresponding programming language below:

[Android SDK](#)

[C SDK](#)

[C++ SDK](#)

[.NET SDK](#)

[Go SDK](#)

[iOS SDK](#)

[Java SDK](#)

[Node.js SDK](#)

[PHP SDK](#)

[Python SDK](#)

[SDK for WeChat Mini Program](#)

Deleting a Bucket

Last updated : 2024-03-25 15:28:24

You can delete a bucket using the console, APIs, tools, or SDKs.

Note:

A bucket cannot be recovered once deleted.

Restrictions

Only empty buckets can be deleted. If there are still objects or incomplete multipart uploads in a bucket, its deletion will fail. Make sure that there are no objects in the bucket before deleting it. For more information, see [Emptying a Bucket](#).

Before deleting a bucket, make sure that the current identity has been granted the permission to delete it and that the correct `Bucket` and `Region` parameters are passed in.

Directions

Using COS console

Delete a bucket in the COS console. For more information, see [Deleting a Bucket](#) in Console Guide.

Using tools

Use tools such as COSBrowser and COSCMD to delete a bucket. For more information, see [Tool Overview](#).

Using REST APIs

Use the REST API to initiate a bucket deleting request. For more information, see [DELETE Bucket](#) in API documentation.

Using SDKs

Directly call the bucket deleting method in the SDK. For more information, see the SDK documentation for the corresponding programming language below:

[SDK for Android](#)

[SDK for C](#)

[SDK for C++](#)

[SDK for .NET](#)

[SDK for Go](#)

[SDK for iOS](#)

[SDK for Java](#)

[SDK for JavaScript](#)

[SDK for Node.js](#)

[SDK for PHP](#)

[SDK for Python](#)

[SDK for Mini Program](#)

Object

Object Overview

Last updated : 2024-03-25 15:28:24

Overview

Object is the basic unit of storage in COS, which can be understood as data in any format, such as images, documents, and audio/video files. Bucket is the carrier of objects. One bucket can contain any number of objects. Different storage classes can be specified for objects in COS. For more information, see [Storage Class Overview](#).

Each object consists of an object key (ObjectKey), an object value (Value), and object metadata (Metadata).

ObjectKey: The unique identifier of an object in a bucket, which can be regarded as the file path. In the API and SDK samples, objects are named in the format of `<ObjectKey>`.

Value: The uploaded object itself, which can be commonly understood as the object content.

Metadata: A set of key-value pairs, which can be regarded as file attributes, such as file modification time and storage class. You can query them after uploading the object.

Object Key

An object in COS must contain a valid `ObjectKey` as the unique identifier of the object in a bucket.

For example, in an object's access address `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/folder/picture.jpg`, the object key is `folder/picture.jpg`.

Naming conventions

You can use any UTF-8 characters in an object key name. However, we recommend you use letters and digits to ensure optimal compatibility with other applications..

The encoded length can be up to 850 bytes.

The name cannot start with a forward slash `/` or backslash `\`.

An object key cannot contain certain ASCII control characters, including upward arrow (↑), downward arrow (↓), rightward arrow (→), and leftward arrow (←), corresponding to CAN (24), EM (25), SUB (26), and ESC (27) respectively.

Refrain from using special symbols such as `*` and `%` in the filename.

Note:

If the name of the uploaded file or folder contains Chinese characters, when you access or request the file or folder, the Chinese characters will be converted into a percent-encoded string based on URL-encoding rules.

For example, when you access `XXX.doc`, the object key is `XXX.doc`, but the percent-encoded string actually read is `%e6%96%87%e6%a1%a3.doc`.

The following are examples of valid object key names:

`doc/exampleobject`

`my.great_photos-2016/01/me.jpg`

`videos/2016/birthday/video.wmv`

Special symbols

Certain characters may need to be URL-encoded or referenced in the hexadecimal format. Some of them are non-printable characters, and your browser may not be able to handle them. They require special handling as detailed below:

,	:	;	=
&	\$	@	+
?	ASCII character ranges: 00–1F hexadecimal (0–31 decimal) and 7F (127 decimal)		(space)

There are also some characters that require significant special handling to maintain consistency across all applications, so we recommend you avoid them directly as detailed below:

`	^	"	\\
{	}	[]
~	%	#	
*	>	<	ASCII 128–255 decimal

Object access address

The access address of an object consists of a bucket access address and an object key in the format of `<bucket domain name>/<object key>`.

For example, after you upload the `exampleobject.txt` object to the `examplebucket-1250000000` bucket in Guangzhou (South China) region, the access address for `exampleobject.txt` is `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/exampleobject.txt`.

Folder and directory

As COS comes with no folders or directories, it will not create a `project` folder for uploading the object `project/a.txt`. To make it easier for you to get started, COS simulates the display mode of "folder" or "directory" in the console and graphical tools such as COSBrowser. This is implemented by creating an empty object with a key value of `project/` and displaying it as a traditional folder.

For example, when you upload the object `project/doc/a.txt` via APIs or SDKs, the separator `/` simulates the display mode of "folder", and you can see the folders `project` and `doc` in the console. The folder `doc` is displayed under the folder `project` and contains the file `a.txt`.

Note:

Objects in the bucket are evenly distributed among distributed clusters. Therefore, you cannot directly get the size of all objects with a specified object key prefix. Instead, you can accumulate the size of each object to get the full size.

Deleting folders and directories is relatively complicated as detailed below:

Deletion Method	Object or Folder Deletion	Result
Console	Folder <code>project</code>	All objects with the object key prefix <code>project/</code> will be deleted.
Console	Object <code>project/doc/a.txt</code>	The <code>project</code> and <code>doc</code> folders will be retained.
API and SDK	Object <code>project/</code> or <code>project/doc/</code>	The object <code>project/doc/a.txt</code> will be retained. If you want to delete the objects in the folder, use code traversal to delete them.

Object metadata

Object metadata (or HTTP header) is a set of key-value pairs in an object. It is the string sent by the server before the server sends HTML data over the HTTP protocol to the browser. Modifying the HTTP header when uploading an object can alter page response forms or communicate configuration information, such as modifying caching time.

There are two types of metadata: system metadata and user-defined metadata.

Note:

Modifying an object's HTTP header does not modify the object itself.

System metadata

This refers to the attribute information of an object, such as object modification time, object size, and storage class.

Name	Description
Content-Length	The length of the content of an HTTP request in bytes as defined in RFC 2616
Last-	The creation date or last modified date of an object (whichever is later)

Modified	
x-cos-version-id	Object version ID. If versioning is enabled for a bucket, the version ID of the object will be returned to identify the historical versions of the object.
ETag	The MD5 value of an object uploaded by calling `PUT Object`, or the unique ID of an object uploaded by multipart upload or a legacy API (which cannot perform check though).

User-defined metadata

This refers to the object's custom parameters, such as Content-Type, Cache-Control, Expires, and x-cos-meta-*. For more information, see [Custom Headers](#).

Name	Description
Cache-Control	The caching policy as defined in RFC 2616, which is saved as part of the object metadata.
Content-Disposition, Content-Encoding, Content-Type	The filename/encoding format/content type (MIME) as defined in RFC 2616, which are saved as part of the object metadata.
Expires	The cache expiration time as defined in RFC 2616, which is saved as part of the object metadata.
x-cos-acl	ACL attributes of an object. Valid values: `private`, `public-read-write`, `public-read`. Default value: `private`.
x-cos-grant-*	Grants the grantee permissions.
x-cos-meta-*	User-defined header, which is returned as part of the object metadata with a size up to 2 KB.
x-cos-storage-class	Storage class of an object. Enumerated values: `STANDARD`, `STANDARD_IA`, `ARCHIVE`. Default value: `STANDARD`.
x-cos-server-side-encryption	Specifies whether to enable server-side encryption for the object and the encryption method.

Object Operations

You can manage objects through the Tencent Cloud console, tools, APIs, and SDKs.

Note:

Objects can be uploaded via [simple upload](#) or [multipart upload](#) based on the object size.

Use simple upload for objects less than 5 GB in size.

Multipart upload is limited to no more than 10,000 parts (each in 5 GB) and a maximum object size of around 48.82 TB. For more information on use limits, see [Specifications and Limits](#).

After uploading objects, you can configure them in the console. For more information, see:

[Searching for Objects](#)

[Viewing Object Information](#)

[Setting Object Access Permission](#)

[Custom Headers](#)

Object Subresource

COS has subresources that are associated with buckets and objects. Subresources are subordinates to objects, so they do not exist on their own. They are always associated with other entities such as objects or buckets. An ACL is the access control information list for a specific COS object, which is a subresource of the object.

An ACL contains an authorization list that identifies authorized users and the granted permissions to implement access control on the object. When an object is created, the ACL identifies the object owner who can fully control the object. You can search for the object ACL or replace it with a new one.

Note:

You can update an ACL only by replacing it.

Access Permission Types

COS supports setting two permissions for objects: **public permissions** and **user permissions**.

Public permissions: Include **Inherit**, **Private Read/Write**, and **Public Read/Private Write**.

Inherit: The object inherits the access permission of the bucket. When you access an object with the inherited bucket permission, COS will match the bucket permission to respond to the access request. A new object inherits the permission from its bucket by default.

Private Read/Write: When you access an object with the private read/write permission, the object can only be accessed with a [request signature](#), regardless of the bucket access permission.

Public Read/Private Write: When you access an object with the public read permission, the object can be directly downloaded, regardless of the bucket access permission.

User permissions: A root account has all the permissions (full access) of buckets by default. In addition, you can add sub-accounts and grant them permissions to read/write data and permissions and even get full access.

Use cases

Allow public access to a specified object in a private read/write bucket, or require authentication for a specific object in a public read/write bucket.

Storage Class

Storage Class Overview

Last updated : 2024-03-25 15:28:24

Cloud Object Storage (COS) offers the MAZ_STANDARD, MAZ_STANDARD_IA, MAZ_INTELLIGENT_TIERING, INTELLIGENT_TIERING, STANDARD, STANDARD_IA, ARCHIVE, and DEEP_ARCHIVE storage classes for objects with different access frequency and disaster recovery levels. They indicate how active objects are in COS, and vary from one another in access frequency, durability, availability, latency, and more. You can choose which storage class to upload your objects to as needed.

Note:

If you did not specify the storage class when uploading an object, it will be uploaded to **STANDARD** by default. For more information on multi-AZ, see [Overview of Multi-AZ Feature](#).

MAZ_STANDARD/STANDARD

Both MAZ_STANDARD and STANDARD storage classes are highly reliable, available, and powerful object storage service designed for hot data and feature low latency and high throughput.

MAZ_STANDARD has higher data durability and service availability than STANDARD. It uses a different storage mechanism to store the data in different data centers in the same region, so as to prevent failures in one data center from affecting the entire service and further guarantee your business stability.

Use cases

Use cases involving lots of hotspot files or frequent data access, including trending videos, social images, mobile apps, game programs, and static websites.

The STANDARD storage class is designed for general use and covers most use cases. It is more cost-effective than MAZ_STANDARD.

However, MAZ_STANDARD has higher data durability and service availability, making it suitable for business scenarios with higher requirements, including key files, commercial data, and sensitive information.

MAZ_STANDARD_IA/STANDARD_IA

Both MAZ_STANDARD_IA and STANDARD_IA storage classes are highly reliable object storage services with low storage costs and access latency. They allow you to access the first byte in milliseconds at a reduced price, so you can retrieve data quickly without waiting. Unlike STANDARD, they involve data retrieval fees when you access data. MAZ_STANDARD_IA uses a different storage mechanism from STANDARD_IA to store the data in different data centers in the same region, so as to prevent failures in one data center from affecting the entire service and further

guarantee your business stability.

Use cases

Use cases with low access frequency (for example, 1 to 2 times per month), such as cloud disk data, big data analysis, government and enterprise data, low-frequency archives, and monitoring data.

Note:

Both MAZ_STANDARD_IA and STANDARD_IA have minimum storage requirements. If the storage duration is less than 30 days, the bill is calculated as 30 days. Likewise, if the size of a file is smaller than 64 KB, the bill is calculated as 64 KB (if the size of a file is greater than or equal to 64 KB, the bill is calculated based on the actual file size). For more information, see [Pricing | Cloud Object Storage](#).

MAZ_INTELLIGENT TIERING/INTELLIGENT TIERING

Objects in the MAZ_INTELLIGENT TIERING storage class can be stored in two storage layers: MAZ_STANDARD and MAZ_STANDARD_IA. Objects in the INTELLIGENT_TIERING storage class can also be stored in two storage classes: STANDARD and STANDARD_IA. COS will automatically switch between storage classes based on the access frequency of such objects with no data retrieval fees incurred, which reduces your storage costs. For more information, see [INTELLIGENT TIERING Overview](#).

MAZ_INTELLIGENT TIERING uses a different storage mechanism from INTELLIGENT TIERING to store the data in different data centers in the same region, so as to prevent failures in one data center from affecting the entire service and further guarantee your business stability.

Use cases

Use cases with uncertain data access patterns. If your business has tight controls on costs and is less sensitive to file reading performance, you can use MAZ_INTELLIGENT TIERING or INTELLIGENT TIERING to reduce costs.

Note:

For MAZ_INTELLIGENT TIERING and INTELLIGENT_TIERING, objects are billed based on their actual sizes. For more information on pricing, see [Pricing | Cloud Object Storage](#).

ARCHIVE

COS ARCHIVE is a highly reliable object storage service designed for cold data, and offers very low storage costs and long-term data retention. This storage class has a minimum storage duration of 90 days. To read data stored in ARCHIVE, you need to restore it to STANDARD first.

COS supports the following three restoration modes for ARCHIVE:

Expedited: Restores an object within 1-5 minutes.

Standard: Restores an object within 3-5 hours.

Bulk: Restores objects within 5-12 hours.

Note:

For more information about object restoration, see [Restoring Archived Objects](#).

The QPS for restoration requests is limited to 100.

Use cases

Use cases that require long-term data retention, such as archival data, medical images, scientific data and other compliance files, lifecycle files, logs, and remote disaster recovery.

Note:

ARCHIVE has minimum storage requirements. If the storage duration is less than 90 days, the bill is calculated as 90 days. Likewise, if the size of a file is smaller than 64 KB, the bill is calculated as 64 KB (if the size of a file is greater than or equal to 64 KB, the bill is calculated based on the actual file size). For more information, see [Pricing | Cloud Object Storage](#).

DEEP ARCHIVE

COS DEEP ARCHIVE is a highly reliable object storage service that offers the lowest storage costs and long-term data retention. This storage class has a minimum storage duration of 180 days. To read data stored in DEEP ARCHIVE, you need to restore it to STANDARD first. For more information, see [Overview - DEEP ARCHIVE](#).

COS supports the following two restoration modes for DEEP ARCHIVE:

Standard: Retrieves an object within 12-24 hours.

Bulk: Retrieves multiple objects within 24-48 hours.

Note:

The QPS of data restoration requests is limited to 100.

Use cases

Use cases that require long-term data retention, such as medical images, view data, and logs.

Note:

DEEP ARCHIVE has minimum storage requirements. If the storage duration is less than 180 days, the bill is calculated as 180 days. Likewise, if the size of a file is smaller than 64 KB, the bill is calculated as 64 KB (if the size of a file is greater than or equal to 64 KB, the bill is calculated based on the actual file size). For more information, see [Pricing | Cloud Object Storage](#).

Storage Class Comparison

Comparison Item	MAZ_STANDARD	MAZ_STANDARD_IA	MAZ_INTELLIGENT_TIERING	INTELLIGENT_TIERING
Storage class	MAZ_STANDARD	MAZ_STANDARD_IA	MAZ_INTELLIGENT_TIERING	INTELLIGENT_TIERING

parameter				
Data durability	99.999999 999%	99.9999999 999%	99.9999999 999%	99.9999999 99%
Service availability	99.995%	99.995%	99.995%	99.99%
Response	Milliseconds	Milliseconds	Milliseconds	Milliseconds
Minimum billable object size	Measured by actual object size	64 KB	Measured by actual object size	Measured by object size
Minimum storage duration	No limit	30 days	No limit	No limit
Supported regions	Only Beijing, Shanghai, Guangzhou, Hong Kong (China), and Singapore	Only Beijing, Shanghai, Guangzhou, Hong Kong (China), and Singapore	Only Beijing, Shanghai, Guangzhou, and Singapore	Only Beijing, Shanghai, Guangzhou, Chengdu, and Tokyo, and Singapore

Storage fees	High	High	Varied by the storage class after intelligent tiering	Varied by the class after intelligent tiering
Data retrieval fees	None	Rather low. Charged by the actual amount of data read	None	None
Request fees	Standard	Rather high	Rather high. You will also be charged the INTELLIGENT TIERING object monitoring fees	Rather high. You will also be charged the INTELLIGENT TIERING object monitoring fees
Data processing	Supported	Supported	Supported	Supported

Storage Class Transition

COS offers various storage classes to indicate how active objects are in COS. You can still change the storage class for objects as needed, or transition objects to a less active storage class such as STANDARD_IA, ARCHIVE, or DEEP ARCHIVE.

Note:
When transitioning objects, ensure that the target storage class is supported for the region where your object resides. To modify the storage class of an object stored in **ARCHIVE** or **DEEP ARCHIVE**, you need to restore it first into STANDARD. For more information, see [Restoring Archived Objects](#).

How each storage class can be transitioned is described as follows:

--	--	--

Storage Class	Available Target Storage Class	Transition Priority
MAZ_STANDARD	MAZ_STANDARD_IA, MAZ_INTELLIGENT TIERING	MAZ_STANDARD > MAZ_STANDARD_IA > MAZ_INTELLIGENT TIERING
MAZ_STANDARD_IA	MAZ_STANDARD, MAZ_INTELLIGENT TIERING	MAZ_STANDARD_IA > MAZ_INTELLIGENT TIERING
MAZ_INTELLIGENT TIERING	MAZ_STANDARD, MAZ_STANDARD_IA	None
INTELLIGENT TIERING	STANDARD, STANDARD_IA, ARCHIVE, DEEP ARCHIVE	INTELLIGENT TIERING > ARCHIVE > DEEP ARCHIVE
STANDARD	INTELLIGENT TIERING, STANDARD_IA, ARCHIVE, DEEP ARCHIVE	STANDARD > STANDARD_IA > INTELLIGENT TIERING > ARCHIVE > DEEP ARCHIVE
STANDARD_IA	INTELLIGENT TIERING, STANDARD, ARCHIVE, DEEP ARCHIVE	STANDARD_IA > INTELLIGENT TIERING > ARCHIVE > DEEP ARCHIVE
ARCHIVE	INTELLIGENT TIERING, STANDARD, STANDARD_IA, DEEP ARCHIVE (after restoration)	ARCHIVE > DEEP ARCHIVE
DEEP ARCHIVE	INTELLIGENT TIERING, STANDARD, STANDARD_IA, ARCHIVE (after restoration)	None

For detailed directions, see the following documents:

[Modifying Storage Classes](#)

[Setting Lifecycles](#)

DEEP ARCHIVE Overview

Last updated : 2024-03-25 15:28:24

Overview

DEEP ARCHIVE is a storage class provided by COS that enables long-term archiving of massive amounts of data. Its unit storage price is comparable to that of tape storage, making it an ideal low-cost solution for long-term data storage. You can manage data easily at low costs through various interaction means provided by COS, such as API, SDK, tools, and console. This eliminates your needs to maintain complex tape library configuration locally and care about the evolution of underlying storage media.

DEEP ARCHIVE is suitable for data that is accessed infrequently but needs to be retained for a long time. In log cold backup scenarios, enterprises need to back up and store the log data generated every day for tracking and analysis according to applicable laws and regulations. In view data and autonomous driving businesses, enterprises accumulate a high number of media files such as images and videos, which need to be archived and stored persistently after use. DEEP ARCHIVE allows enterprises to store such data in the cloud and restore it only when needed, which reduces the storage costs and simplifies Ops management.

DEEP ARCHIVE is designed for 99.999999999% durability and 99.95% availability. You can also use the versioning and cross-region replication features provided by COS to further guarantee the data security.

How to Use

1. Upload using the console.

1. Log in to the [COS console](#).

2. Click **Bucket List** and create a bucket in a supported region (such as Beijing).

3. After creating the bucket, click **Upload File** in **File List**.

4. Select a local file for upload and select **DEEP ARCHIVE** for **Storage Class** in **Set Properties**. For more information, see [Uploading Objects](#).

✓ Select Objects

>

2 Set Properties

Properties Setting will be applied to all the objects to be uploaded, you can also upload directly and then modify the settings in file list page.

Storage Class

☒ STANDARD
It is suitable for business scenarios such as real-time access to a large number of hot files and frequent data interaction. Supported in all regions.
☐ STANDARD_IA
It is suitable for business scenarios with low access frequency (e.g., average access frequency is 1 to 2 times per month). Supported in all regions.
☐ ARCHIVE
It is suitable for business scenarios with very low access frequency (e.g., once every six months). Since real-time response is not supported, if you want to retrieve the archived data, please apply in advance.
☐ Deep Archive Storage
It is suitable for business scenarios with very low access frequency (for example, 1 to 2 visits per year). If you want to retrieve the data stored in deep archives, you need to apply in advance and cannot respond in real time.

Access Permissions

☒ Inherit
☐ Private Read/Write
☐ Public Read/Private Write

Server-Side Encryption

☒ None
☐ SSE-COS ⓘ
☐ SSE-KMS

Click here to use latest KMS service.[here to activate it](#).

Object Tag

+

Metadata

Parameter	Value	Operation
Select Project ▼	Value	Delete
Add Parameters		

5. Upload using APIs.

Set `x-cos-storage-class` to `DEEP_ARCHIVE` in the [PUT Object](#), [POST Object](#), or [Initiate Multipart Upload](#) API for direct upload to the DEEP ARCHIVE storage class.

Note:

`Append Object` does not support direct upload to DEEP ARCHIVE.

6. Upload using SDKs.

Currently, all the SDKs released by COS support direct upload to DEEP ARCHIVE. You need to set the

`StorageClass` parameter to `DEEP_ARCHIVE` when uploading a file.

7. Upload using a tool.

COSBrowser and COSCMD support direct upload to DEEP ARCHIVE. You need to add the header field `x-cos-`

`storage-class` and set it to `DEEP_ARCHIVE` when uploading a file.

Use Limits

DEEP ARCHIVE is subject to the following limits. When using it, you need to pay attention to their impact on the storage costs and performance:

Minimum storage unit: The minimum storage unit is 64 KB, and objects smaller than 64 KB will be counted as 64 KB during the calculation of the storage capacity.

Minimum storage period: The minimum storage period is 180 days, and a shorter period will be counted as 180 days for billing.

Note:

The storage period means logical days (a day is 24 hours) and starts from the time when a file is modified.

Retrieval request QPS limit: 100 request/sec.

Available regions: DEEP ARCHIVE is currently supported only in the Beijing, Nanjing, Shanghai, Guangzhou, Chengdu, Chongqing, Tokyo, and Singapore regions and will be available in more regions.

Use limit: Currently, DEEP ARCHIVE is not supported for multi-AZ buckets.

Operation limit: Objects cannot be appended to other objects in the DEEP ARCHIVE storage class.

Retrieval request limit: Only one retrieval request can be executed at a time for an object, and multiple repeated requests will be merged and processed according to the fastest retrieval mode. If the retrieval mode of the (N+1)th request is faster than that of the Nth request, the (N+1)th request will be executed as a new request; otherwise, it will fail.

Data retrieval: Standard retrieval (12–24 hours) and bulk retrieval (24–48 hours) are supported for files in the DEEP ARCHIVE storage class.

Overview - INTELLIGENT TIERING

Last updated : 2024-06-25 15:35:05

Overview

INTELLIGENT TIERING is a COS storage class designed to reduce storage costs by automatically moving objects between two storage tiers (frequent access and infrequent access) when access patterns change.

INTELLIGENT TIERING is suitable for data with an unpredictable or variable access pattern. COS monitors the access situation of objects, and storage fees are charged based on the actual storage tier utilized (frequent access tier, infrequent access tier, archive tier, and deep archive tier). Users can change the storage class of objects with uncertain access patterns from STANDARD to INTELLIGENT TIERING as needed to reduce in-cloud storage costs.

Note:

INTELLIGENT TIERING is currently available only in the Beijing, Nanjing, Shanghai, Guangzhou, Chengdu, Chongqing, Singapore, Mumbai, Tokyo, and Frankfurt regions.

INTELLIGENT TIERING is a standalone storage class that will incur storage usage fees and object monitoring fees. You can purchase an INTELLIGENT TIERING storage pack to redeem the storage usage fees. For detailed pricing, see [Product Pricing](#).

MAZ buckets do not support enabling configurations for intelligent tiering archive/deep archive tiers.

Advantages

The access patterns of data stored in INTELLIGENT TIERING will be monitored. COS moves objects that have not been accessed for a continuous period to the lower-cost infrequent access tier. If an object in the infrequent access tier is accessed later, it is automatically moved back to the frequent access tier to ensure the data retrieval performance. INTELLIGENT TIERING manages to strike a balance between storage costs and read/write performance and features the following strengths:

Cost-effective: When you use INTELLIGENT TIERING for persistent storage, the longer your data is stored, the more you can save on your storage costs (up to about 20%) compared with the STANDARD storage class.

INTELLIGENT TIERING is also involved in the object storage lifecycle, where you can transition your INTELLIGENT TIERING data to ARCHIVE to further lower storage costs.

Stable and durable: INTELLIGENT TIERING provides the same low latency and high throughput as the STANDARD storage class. Besides, it offers durability of up to 99.999999999% (11 9s) by using erasure code-based redundancy, and up to 99.99% availability by using block storage and concurrent reads/writes. INTELLIGENT TIERING has also been integrated with the COS MAZ architecture, featuring high durability (99.999999999%, that is, 12 9s) and availability (99.995%).

Easy-to-use: To use INTELLIGENT TIERING, all you need to do is specify it as the storage class of your object. This COS storage class is inherently compatible with COS APIs, SDKs, tools, and applications, making it easy for you to manage your in-cloud data as needed.

Supported Storage Tiers

Introduction to Storage Tiers

Note:

Files smaller than 64KB will remain in the standard tier and will not descend to the infrequent access tier, archive tier, or deep archive tier.

Users can check which tier an intelligent tiering object is in by using the header `x-cos-storage-tier` returned by the API [HeadObject](#). Additionally, in the list of objects returned by calling APIs `GetBucket` and `GetBucketVersions`, an intelligent tiering object will include the `StorageTier` field for viewing the object's tier.

Access Tier	x-cos-storage-tier
Frequent Access Tier	FREQUENT
Infrequent Access Tier	INFREQUENT
Archive Tier	ARCHIVE_ACCESS
Deep Archive Tier	DEEP_ARCHIVE_ACCESS

Frequent Access Tier (Enabled by default)

After INT objects are uploaded, they are in the frequent access tier (FREQUENT) by default. While objects are in this tier, storage fees are billed according to the regional standard storage list price.

Infrequent Access Tier (Enabled by default)

When enabling intelligent tiering for a bucket, you need to select the number of days after which to transition to the infrequent access tier, with options including 30, 60, and 90 days. Once set, it cannot be modified.

If an object remains unaccessed for a continuous period of 30 days (or 60 or 90 days), it will transition from the frequent access tier to the infrequent access tier (INFREQUENT). While the object is in this tier, storage fees are billed according to the current region's infrequent storage list price. After being accessed, the object will return to the frequent access tier.

Archive Tier (Optional)

Since objects in the archive tier must be restored before access, enabling the archive tier is optional. Users can enable the archive tier for objects with specific prefixes or tags by adding one or more intelligent tiering archive

configuration rules, and set the transition days. The minimum transition period of the archive tier is 91 days, and the maximum is 730 days.

Note:

If the current region does not support archive storage, the console will not allow adding archive configuration rules, and intelligent tiering objects will not descend to the archive tier.

When intelligent tiering objects are in the archive tier, storage fees are charged according to the regional archive storage list price, and there will be no early deletion fees.

Restoring Archive Tier objects

When an object remains unaccessed for N consecutive days, it will transition from the infrequent access tier to the archive tier. Once in the archive tier, data can only be read after the object is restored via [PostObjectRestore](#).

Unlike regular archive types, restoring an object from an INT archive tier will not create a standard-type replica. Instead, the object itself will directly return to the frequent access tier.

Similar to the regular archive type, INT archive tier objects support three retrieval modes: expedited, standard, and bulk. **In terms of fees, only expedited retrieval fees are charged, and the list price is the same as the regional archive storage expedited retrieval fees. Retrieval request fees for all modes, and retrieval fees for standard and bulk modes are not charged.**

Deep Archive Tier (Optional)

Similarly, deep archive tier objects need to be restored before access. Therefore, enabling the deep archive tier is optional. Users can activate the deep archive tier for objects with specific prefixes or tags by adding one or more intelligent tiering deep archive configuration rules. It is possible to configure both the archive tier and deep archive tier transition in the same rule. It is important to note that the transition days for the deep archive tier must be at least 180 days and a maximum of 730 days, **and must be greater than the transition days for the archive tier.**

Note:

If the current region does not support deep archive storage, the console will not allow adding deep archive configuration rules, and intelligent tiering objects will not descend to the deep archive tier.

When INT objects are in the deep archive tier, storage fees are charged according to the regional deep archive storage list price, and there will be no early deletion fees.

Restoring objects from the Deep Archive Tier

When an object remains unaccessed for M days, it will transition from the infrequent access tier/archive tier to the deep archive tier. Once in the deep archive tier, data can be read only after the object is restored through [PostObjectRestore](#).

Unlike regular deep archive types, restoring an object from the INT Deep Archive Tier will not produce a standard-type replica; instead, the object itself will directly return to the frequent access tier.

Similar to regular deep archive types, INT deep archive tier objects support two retrieval modes: standard and bulk, **and there will be neither retrieval fees nor retrieval request fees.**

Enabling Intelligent Tiering, Archive Tier and Deep Archive Tier Configuration

Rule Description

Once a bucket is enabled with intelligent tiering configuration, the uploaded intelligent tiering objects by default only switch between the frequent access tier and the infrequent access tier. After archive and deep archive tier rules are added in the intelligent tiering configuration, switching to the archive and deep archive tiers can be enabled. Each bucket can support up to 1000 archive and deep archive tier configuration rules, with the elements included in the rules as follows. For more details on the rules, refer to the API documentation [PUT Bucket IntelligentTiering](#).

Rule Name (Id)

Used to uniquely identify archive and deep archive rules.

Status

Supports Enabled or Disabled. When in the disabled state, even if archive/deep archive tier rules are configured, transition to the archive or deep archive tiers will not actually occur.

Application Scope (Filter)

The effective range of the rules. It supports prefix filtering and tag filtering. The number of prefixes cannot exceed one, and the number of tags cannot exceed ten.

Hierarchy Settings

Supports setting the transition time for the archive tier and deep archive tier within a single rule. After the corresponding rules are set, INT objects will start the transition to the archive and deep archive tiers. Otherwise, they will only switch between the frequent access tier and the infrequent access tier.

Archive Tier (ARCHIVE_ACCESS): Supports the setting between 91 to 730 days.

Deep Archive Tier (DEEP_ARCHIVE_ACCESS): Supports the setting between 180 to 730 days.

Storage Tier Transition Order

Users can enable archive tier transition and deep archive tier transition separately or enable both at the same time. The transition will proceed sequentially based on the configured days for the infrequent access tier, archive tier, and deep archive tier.

Configuration Samples	Configuration Details	Effective Result
Sample 1	30 days for the infrequent access tier, 100 days for the archive tier, and 190 days for the deep archive tier	After 30 consecutive days of no access, transition from the standard access tier to the infrequent access tier will occur.

		<p>After 100 consecutive days of no access, transition from the infrequent access tier to the archive tier will occur.</p> <p>After 190 consecutive days of no access, transition from the archive tier to the deep archive tier will occur.</p>
Example 2	30 days for the infrequent access tier, and 190 days for deep archive tier	<p>After 30 consecutive days of no access, transition from the standard access tier to the infrequent access tier will occur.</p> <p>After 190 consecutive days of no access, transition from the infrequent access tier to the deep archive tier will occur.</p>
Example 3	60 days for the infrequent access tier and 91 days for the archive tier	<p>After 60 consecutive days of no access, transition from the standard access tier to the infrequent access tier will occur.</p> <p>After 91 consecutive days of no access, transitions from the infrequent access tier to the archive tier will occur.</p>

Restoring INT Objects from the Archive and Deep Archive Tiers

When an object remains unaccessed for many consecutive days, it will be transitioned from the infrequent access tier to the archive/deep archive tier. Once it enters the archive or deep archive tier, it needs to be restored using [PostObjectRestore](#) before the data can be read.

Unlike the restoration of regular archive types and deep archive types of objects, the restoration of INT archive and deep archive tier objects **does not create a standard replica; instead, the object itself will directly return to the frequent access tier. Therefore, when initiating a retrieval request for INT archive/deep archive tier objects, there is no need to specify the number of days for retrieval.**

Similar to the regular archive types, INT archive tier objects support fast, standard, and bulk retrieval modes; like the regular deep archive types, INT deep archive tier objects support standard and bulk retrieval modes.

In terms of charges, apart from the archive tier fast retrieval fee (priced the same as the regional archive storage fast retrieval fee), no other retrieval fees, retrieval request fees, or rewarming replica storage fees are charged.

	Archive Storage Type	Intelligent Tiering Object - Archive Tier	Deep Archive Storage Type	Intelligent Tiering Object - Deep Archive Tier
Storage Fee	Storage fees are charged according to the current regional archive storage type		Storage fees are charged according to the current regional deep archive storage type	

Whether retrieval generates a replica		Yes, it is necessary to specify the retention period, and replica storage fees are incurred according to standard storage pricing.	No, there will be no replica storage fees.	Yes, it is necessary to specify the retention period, and replica storage fees are incurred according to standard storage pricing.	No, there will be no replica storage fees.
Retrieval Mode		Expedited mode Standard mode Batch mode	Expedited mode Standard mode Batch mode	Standard mode Batch mode	Standard mode Batch mode
Retrieval Fee (In USD/GB)	Fast	Fast Retrieval Fee for Archiving	Fast Retrieval Fee for Intelligent Tiering Archive Tier (Same Pricing as Archiving Storage)	\\	\\
	Standard	Standard Retrieval Fee for Archiving	Free	Deep Archiving Standard Retrieval Fee	Free
	Batch	Archival Batch Retrieval Fee	Free	Deep Archive Batch Retrieval Fee	Free
Retrieval Request Fees (In USD/10,000 Calls)		Free	Free	Deep Archive Standard Retrieval Request Fee Deep Archive Bulk Retrieval Request Fee	Free
Read/Write Request Fee		Consistent with the standard storage fee after the restoration	Intelligent Tiering Read and Write Request Fee	Deep Archive Read and Write Request Fee	Intelligent Tiering Read and Write Request Fee

During the restoration process, you can use HeadObject to check the restoration status of INT archive tier/deep archive tier objects.

When in the process of restoration, the response header of HEAD Object will include x-cos-restore and x-cos-restore-status. For example, x-cos-restore:ongoing-request="true", cos-restore-status:tier="bulk"; request-date="Mon, 18 Nov

2019 09:34:50 GMT".

After the restoration is complete, INT objects directly return to the frequent access tier, and the corresponding head part of HEAD Object x-cos-storage-tier is FREQUENT.

How to Use

To store your object in the COS INTELLIGENT TIERING storage class, enable INTELLIGENT TIERING for the bucket, and then specify the **storage class** of your object as INTELLIGENT TIERING.

Using COS console

Setting INTELLIGENT TIERING upon object uploads

You can perform the following steps to store your object in the INTELLIGENT TIERING storage class:

1. On the bucket configuration page, enable INTELLIGENT TIERING. For more information on the process of enabling INTELLIGENT TIERING, see [Setting INTELLIGENT TIERING](#).
2. Upload a file and specify the storage class during the upload. For more information about how to upload a file, see [Uploading Objects](#).

Note:

Once INTELLIGENT TIERING is enabled for a bucket, it cannot be disabled.

Moving in-cloud objects to INTELLIGENT TIERING

You can perform the following steps to move in-cloud objects to the INTELLIGENT TIERING storage class:

1. On the bucket configuration page, create a lifecycle rule. For detailed directions, see [Setting Lifecycle](#).
2. Specify the application range and transition objects to INTELLIGENT TIERING.

Enabling Intelligent Tiering Archive and Deep Archive Tier

Users can enable intelligent tiering archive and deep archive tier configuration for buckets as needed. For details, see [Setting Intelligent Tiering Storage](#) documentation.

Using the REST API

You can use the following APIs to configure INTELLIGENT TIERING:

1. Use RESTful APIs to enable INTELLIGENT TIERING for your bucket. For more information, see the following API documentation:

To enable intelligent tiering and set the transition days for the infrequent access tier:

[PUT Bucket IntelligentTiering](#) (id is not default)

[GET Bucket IntelligentTiering](#) (id is not default)

To configure or delete archive and deep archive tier rules:

[PUT Bucket IntelligentTiering](#) (id is not default)

[GET Bucket IntelligentTiering](#) (id is not default)

[Delete Bucket IntelligentTiering](#) (id is not default)

2. After enabling INTELLIGENT TIERING for the bucket, use the following APIs to store the object in the INTELLIGENT TIERING storage class:

[PUT Object](#)

[PUT Object - Copy](#)

[POST Object](#)

[Initiate Multipart Upload](#)

3. You can use the following APIs to query the storage class or storage tier of an object:

[GET Object](#)

[HEAD Object](#)

4. You can use the following RESTful APIs to delete objects in the INTELLIGENT TIERING storage class:

[DELETE Object](#)

[DELETE Multiple Objects](#)

[PUT Bucket lifecycle](#)

[Delete Bucket IntelligentTiering](#)

5. To retrieve objects from the intelligent tiering archive tier and deep archive tier, please refer to the following API:

[PostObjectRestore](#)

Using the SDK

Currently, all COS SDKs support INTELLIGENT TIERING. To use this storage class, set `StorageClass` to `INTELLIGENT_TIERING` or `MAZ_INTELLIGENT_TIERING` when uploading an object. For more information, see [Uploading Object](#).

Use Limits

INTELLIGENT TIERING has the following limits:

Setting: The setting cannot be changed once configured. To change it, [contact us](#). The number of consecutive no-access days for moving objects to the infrequent access tier can be set to 30, 60, or 90.

Initial storage tier: A new object is stored in the frequent access tier by default, and will only be moved to the infrequent access tier if it hasn't been accessed for a certain period.

Minimum storage size: An object smaller than 64 KB can only be stored in the frequent access tier and cannot be moved between the frequent and infrequent access tiers. Objects are billed based on their actual sizes.

Operation: Uploading objects to INTELLIGENT TIERING using the APPEND Object API is not supported.

Lifecycle: Objects in the INTELLIGENT TIERING storage class can only be transitioned to ARCHIVE or DEEP ARCHIVE. If a STANDARD object is transitioned to INTELLIGENT TIERING, it will be stored in the frequent access

tier. If a STANDARD_IA object is transitioned to INTELLIGENT TIERING, it will be stored in the infrequent access tier.

Bucket replication: During bucket replication, if INTELLIGENT TIERING is not enabled for the destination bucket, the object cannot be copied to the INTELLIGENT TIERING storage class.

FAQs

How is INTELLIGENT TIERING billed?

INTELLIGENT TIERING fees include **INTELLIGENT TIERING storage usage fees** and **INTELLIGENT TIERING object monitoring fees**.

1. INTELLIGENT TIERING storage usage fees are charged differently depending on the storage tier of objects.

When objects are stored in the frequent access tier, STANDARD storage usage fees are charged.

When objects are stored in the infrequent access tier, STANDARD_IA storage usage fees are charged.

Note:

STANDARD and STANDARD_IA storage usage fees vary by region. For details about their pricing, see [Product Pricing](#).

Request fees and traffic fees are also incurred during object upload and download. For the billing examples, see [Traffic Fee Billing Example](#) and [Request Fee Billing Example](#).

2. INTELLIGENT TIERING object monitoring fees are charged based on the number of objects stored (excluding files smaller than 64 KB). For detailed pricing, see [Pricing | Cloud Object Storage](#).

Example

Suppose an organization has 100,000 objects (all above 64 KB, 1 TB in total), and the data is stored in the INTELLIGENT TIERING storage class in Beijing region and transitioned to the infrequent access tier after 30 days. If 20% of the objects (i.e., 20,000 objects) are transitioned to the infrequent access tier every 30 days, the object monitoring fees and storage usage fees for **every 30 days** will be as follows:

Note:

In the table below, the object monitoring fees for Beijing region are 0.25 USD/10,000 objects per 30 days, that is, 0.00833333 (0.25 / 30) USD/10,000 objects per day.

Storage Days	30-day Object Monitoring Fees (USD)	30-day INTELLIGENT TIERING Storage Usage Fees (USD)	30-day STANDARD Storage Usage Fees (USD)
30 * 1	0.25/10,000 objects * 100,000	$1024 * 0.024 / 30 * 30 = 24.58$	$1024 * 0.024 / 30 * 30 = 24.58$
30 * 2	0.25/10,000 objects * 100,000	$819.2 * 0.024 / 30 * 30 + 204.8 * 0.08 / 30 * 30 = 23.35$	$1024 * 0.024 / 30 * 30 = 24.58$
30 * 3	0.25/10,000 objects * 100,000	$655.36 * 0.024 / 30 * 30 + 368.64 * 0.08 / 30 * 30 = 23.35$	$1024 * 0.024 / 30 * 30 = 24.58$

	100,000	$0.08 / 30 * 30 = 22.36$	
$30 * 4$	$0.25/10,000 \text{ objects} * 100,000$	$524.288 * 0.024 / 30 * 30 + 499.712 * 0.08 / 30 * 30 = 21.58$	$1024 * 0.024 / 30 * 30 = 24.58$
$30 * 5$	$0.25/10,000 \text{ objects} * 100,000$	$419.4304 * 0.024 / 30 * 30 + 604.5696 * 0.08 / 30 * 30 = 20.95$	$1024 * 0.024 / 30 * 30 = 24.58$
$30 * 6$	$0.25/10,000 \text{ objects} * 100,000$	$335.54432 * 0.024 / 30 * 30 + 688.45568 * 0.08 / 30 * 30 = 20.45$	$1024 * 0.024 / 30 * 30 = 24.58$

As you can see, using INTELLIGENT TIERING reduces storage costs over time. Only a small amount of monitoring cost is required every 30 days.

What types of objects is INTELLIGENT TIERING suitable for?

INTELLIGENT TIERING is suitable for large objects (such as audio and video, logs, etc.) whose access patterns change. Larger average object sizes mean you pay less for monitoring per GB objects. If your business has relatively fixed data access patterns, you can set lifecycle configuration to specify the time to transition objects to STANDARD_IA without the need to use INTELLIGENT TIERING.

How can I store objects in INTELLIGENT TIERING?

You can store objects in INTELLIGENT TIERING as follows:

Incremental objects: You just need to specify the storage class as INTELLIGENT TIERING when uploading the objects.

Existing objects: You can modify the storage class to INTELLIGENT TIERING using the `COPY` API. You can also use the lifecycle feature to transition STANDARD or STANDARD_IA objects to INTELLIGENT TIERING.

Note:

In INTELLIGENT TIERING, objects smaller than 64 KB will always be stored in STANDARD. Therefore, for objects smaller than 64 KB, we recommend you upload them to the STANDARD, STANDARD_IA, ARCHIVE, or DEEP ARCHIVE storage class as needed to reduce costs.

How do I disable INTELLIGENT TIERING configuration?

INTELLIGENT TIERING configuration **can't be disabled** once enabled. If you don't need to store your objects in INTELLIGENT TIERING, you just need to specify the storage class as a non-INTELLIGENT TIERING class such as STANDARD, STANDARD_IA, ARCHIVE, or DEEP ARCHIVE when uploading the objects.

The rules for intelligent tiering archive tier/deep archive tier support deletion. You can delete the corresponding rules to prevent new INT objects from descending into the archive tier/deep archive tier.

How are charges applied for archiving in the intelligent tiering and deep archive tier?

Storage fee: Charged according to the regional archive and deep archive storage fees.

Retrieval fee and retrieval request fee: Apart from the fast retrieval fees for the archive tier, no other retrieval fees nor retrieval request fees are charged.

Rewarming replica fee: Not charged.

Uploading Object

Simple Upload

Last updated : 2024-03-25 15:28:24

Simple upload refers to uploading objects using the `PUT Object` API. It is suitable for uploading an object smaller than 5 GB in a single request.

To upload an object larger than 5 GB, you can use:

COS console: You can upload an object up to 512 GB. For more information, see [Uploading Objects](#).

Multipart upload with APIs or SDKs: You can upload an object up to 48.82 TB (i.e., 50,000 GB). For more information, see [Multipart Upload](#).

COSCMD: You can upload an object up to 40 TB. For more information, see [COSCMD](#).

Note:

When initiating a request, if you want to specify a directory or path, you can use `/`. For example, if you need to upload `picture.png` to the `doc` directory, set the object key to `doc/picture.png`.

Use Cases

Simple upload is suitable for uploading an object smaller than 5 GB.

In a high bandwidth or weak network environment, if your object is relatively large, for example, over 100 MB, we recommend you use [Multipart Upload](#), even if the object is smaller than 5 GB. This is because multipart upload supports uploading multiple parts concurrently, which gives full play to resources in high bandwidth environments. On the other hand, in weak network environments, the uploaded parts will not be affected by the failed ones. Therefore, failed parts can be re-uploaded with a simple retry, improving the overall upload success rate. For more information about uploading objects with a client in a weak network environment, see [Multipart Upload Resumption in a Weak Network Environment](#).

Directions

Using REST APIs

Use REST APIs to initiate a simple upload request. For more information, see [PUT Object](#).

Using SDKs

Directly call the simple upload method in the SDK. For more information, see the SDK documentation for the corresponding programming language below:

[Android SDK](#)

[C SDK](#)

[C++ SDK](#)

[.NET SDK](#)

[Go SDK](#)

[iOS SDK](#)

[Java SDK](#)

[JavaScript SDK](#)

[Node.js SDK](#)

[PHP SDK](#)

[Python SDK](#)

[Mini Program SDK](#)

Multipart Upload

Last updated : 2024-03-25 15:28:24

Overview

Multipart upload can split an object into multiple parts and upload them to COS. When uploaded, the parts are numbered consecutively. You can upload each part separately or upload them in any order. COS will merge them into an object based on their numbers. If the upload of any part fails, the failed part can be uploaded again without affecting other parts and the content as a whole. In a poor network environment, multipart upload is recommended for objects larger than 20 MB. In a high-bandwidth environment, multipart upload is recommended for objects larger than 100 MB.

Multipart upload can split an object into up to 10,000 parts of 1 MB to 5 GB in size each. The last part can be smaller than 1 MB.

Note:

Simple upload only supports uploading files with a maximum size of 5 GB, while multipart upload supports larger files.

Use Cases

Multipart upload is suitable for uploading large objects in poor or high-bandwidth network environments.

Multipart upload has the following strengths:

In a poor network environment, smaller part size minimizes the impact of restarting a failed upload due to network errors.

In a high-bandwidth environment, multipart upload can maximize the use of your available bandwidth by uploading object parts in parallel. Uploading parts out of order does not affect the final merged object.

With multipart upload, you can pause and resume the upload of a single large object at any time. All incomplete multipart uploads can be resumed unless aborted.

Multipart upload can also be used to begin an upload before you know the final object size. You can initiate an upload and then merge the parts to get the full object size.

How to Use

Using RESTful API

You can use a RESTful API directly to initiate a multipart upload request. For more information, see the following API documents:

[Initiate Multipart Upload](#)[Upload Part](#)[Complete Multipart Upload](#)[Abort Multipart Upload](#)

Using SDK

You can directly call the multipart upload method in the SDK. For more information, see the SDK documentation for the corresponding programming language below:

[Android SDK](#)[C SDK](#)[C++ SDK](#)[.NET SDK](#)[Go SDK](#)[iOS SDK](#)[Java SDK](#)[JavaScript SDK](#)[Node.js SDK](#)[PHP SDK](#)[Python SDK](#)[Mini Program SDK](#)

Upload via Pre-Signed URL

Last updated : 2024-03-25 15:28:24

Use Cases

All buckets and objects are private by default. If you want any third party to be able to upload an object to your bucket, but you don't want them to use CAM accounts or temporary keys, signatures can be provided by pre-signed URLs for temporary upload operations. Anyone who receives a valid pre-signed URL can upload an object.

When creating a pre-signed URL, you can include an object key in your signature so that the object can only be uploaded to the specified object key. Besides, the validity period of pre-signed URLs can be provided in SDKs to ensure that expired URLs will not be used by any unauthorized party.

Note:

You are advised to use a temporary key to generate a pre-signed URL for the security of your requests such as uploads and downloads. When you apply for a temporary key, follow the [Principle of Least Privilege](#) to avoid leaking resources besides your buckets and objects.

If you need to use a permanent key to generate a pre-signed URL, you are advised to limit the permission of the permanent key to uploads and downloads only to avoid risks.

Directions

Using SDKs

You can call the pre-signed URL method in the SDK directly. For more information, please see the SDK documentation for the corresponding programming language below:

[Android SDK](#)

[C SDK](#)

[C++ SDK](#)

[.NET SDK](#)

[Go SDK](#)

[iOS SDK](#)

[Java SDK](#)

[JavaScript SDK](#)

[Node.js SDK](#)

[PHP SDK](#)

[Python SDK](#)

[Mini Program SDK](#)

Downloading Object

Simple Download

Last updated : 2024-03-25 15:28:24

Use Cases

You can directly initiate a request to download objects in COS. The following features are supported:

Download a complete object: Download the complete object data by initiating a GET request.

Download a part of an object: Use the Range request header in a GET request to retrieve a specific range of bytes of an object. Retrieving multiple ranges is not supported.

The object's metadata will be returned along with the object's content as an HTTP response header. The GET request supports overwriting certain metadata values in the response using URL parameters.

For example, the response value of Content-Disposition can be overwritten. Response headers that support modification include:

Content-Type

Content-Language

Expires

Cache-Control

Content-Disposition

Content-Encoding

How to Use

Using COS console

Download an object in the COS console. For more information, see [Downloading Objects](#) in Console Guide.

Using the REST API

Use the REST API to initiate an object download request. For more information, see [GET Object](#).

Using SDKs

Directly call the object download method in the SDK. For more information, see the SDK documentation for the corresponding programming language below:

[SDK for Android](#)

[C SDK](#)

[C++ SDK](#)

[.NET SDK](#)

[SDK for Go](#)

[SDK for iOS](#)

[SDK for Java](#)

[SDK for JavaScript](#)

[SDK for Node.js](#)

[SDK for PHP](#)

[SDK for Python](#)

[SDK for Weixin Mini Program](#)

Download via Pre-Signed URL

Last updated : 2024-03-25 15:28:24

Overview

All buckets and objects are private by default. If you want any third party to be able to download an object without using CAM account or temporary keys, provide the third parties with signatures via pre-signed URLs for download operations. Anyone who receives a valid pre-signed URL can download an object.

When creating a pre-signed URL, you can include object keys in your signature to specify the objects allowed for download. Besides, the validity period of pre-signed URLs can be provided in SDKs to ensure that expired URLs will not be used by any unauthorized party.

Note:

Accessing a CDN domain needs to follow the authentication process of CDN, in which case COS signatures cannot be used. Therefore, pre-signed URLs do not support the use of CDN domains.

You are advised to use a temporary key to generate pre-signed URLs for the security of your requests such as uploads and downloads. When you apply for a temporary key, follow the [Principle of Least Privilege](#) to avoid leaking resources besides your buckets and objects.

If you need to use a permanent key to generate a pre-signed URL, you are advised to limit the permission of the permanent key only to uploads and downloads to avoid risks. In addition, the validity period of the generated URL must be set to the shortest period required to complete the current upload or download operation. This is because the request will be interrupted when the validity period of the specified pre-signed URL expires. In addition, the checkpoint restart is not supported, and therefore the failed request needs to be re-executed after a new URL is applied for.

Directions

Using SDKs

Call the pre-signed URL method in the SDK. For more information, see the SDK documentation for the corresponding programming language below:

[Android SDK](#)

[C SDK](#)

[C++ SDK](#)

[.NET SDK](#)

[Flutter SDK](#)

[Go SDK](#)

[iOS SDK](#)

[Java SDK](#)

[JavaScript SDK](#)

[Node.js SDK](#)

[PHP SDK](#)

[Python SDK](#)

[React Native SDK](#)

[Mini Program SDK](#)

Listing Object Keys

Last updated : 2024-03-25 15:28:24

An [object key](#) is the unique identifier of an object in a bucket. You can think of it as the object's path. For example, if an object key is `doc/picture.jpg`, the image `picture.jpg` is stored in the `doc` path/folder in COS. You can use the object key to search for a specific object. You can also use a prefix of the object key (e.g. `doc`) to search for all objects with this prefix (e.g. all objects prefixed with `doc`).

Overview

Tencent Cloud COS supports listing keys by a prefix. You can use `/` in a key to implement a hierarchical structure similar to the traditional file system. In COS, you can use a delimiter to select and browse keys hierarchically. You can list all keys in a single bucket in UTF-8 binary order of prefixes or filter the key list by specifying the prefix. For example, adding the parameter `t` will list the `tencent` object, but skip objects prefixed with `a` or other characters.

`/` can be used as a delimiter in object keys. In this way, both the prefix and delimiter can be used to facilitate the search.

COS allows you to store an unlimited number of objects in a single bucket. As a result, the key list may be very large. For the convenience of management, a maximum of 1,000 key values can be returned in each `List Objects` request, and a marker will be returned to indicate whether the list is truncated. If so, not all objects are listed in this request. In this case, you can initiate the `List Objects` request multiple times based on markers and delimiter to list all/some object keys as needed.

Directions

Using COS console

Search for objects in the COS console. For more information, see [Searching for Objects](#) in Console Guide.

Using REST APIs

Use REST APIs to initiate a request to list object keys. For more information, see [GET Bucket \(List Objects\)](#).

Using SDKs

Directly call the object list querying method in the SDK. For more information, see the SDK documentation for the corresponding programming language below:

[SDK for Android](#)

[C SDK](#)

[C++ SDK](#)

[.NET SDK](#)

[Go SDK](#)

[SDK for iOS](#)

[Java SDK](#)

[JavaScript SDK](#)

[Node.js SDK](#)

[PHP SDK](#)

[Python SDK](#)

[SDK for WeChat Mini Program](#)

Copying Objects

Simple Copy

Last updated : 2024-03-25 15:28:24

Use Cases

The copy operation creates a copy of a COS object of up to 5 GB in a single request. To copy an object over 5 GB, use the [multipart copy](#) API. With the copy operation, you can:

Create a copy of an object.

Rename an object by copying it and deleting the original one.

Modify the storage class of an object. You can select the same object key as both the source and target and modify the storage class.

Copy objects across different COS regions.

Modify object metadata. You can select the same object key as both the source and target and modify object metadata.

In the copy operation, the metadata of the source object is inherited by default, while the creation date is subject to the creation date of the target object.

Note:

Copy and paste is not supported for objects in the ARCHIVE storage class.

Objects in MAZ buckets cannot be replicated to an OAZ bucket.

A sub-account should be granted the `PutObject` , `GetObject` , and `GetObjectACL` permissions to copy objects.

How to Use

Using COS console

Copy objects in the COS console. For more information, see [Copying Objects](#) in Console Guide.

Using the REST API

Use REST APIs to initiate an object copy request. For more information, see [PUT Object - Copy](#).

Using SDKs

Directly call the object copy method in the SDK. For more information, see the SDK documentation for the corresponding programming language below:

[SDK for Android](#)

[C SDK](#)

[C++ SDK](#)

[.NET SDK](#)

[SDK for Go](#)

[SDK for iOS](#)

[SDK for Java](#)

[SDK for JavaScript](#)

[Node.js SDK](#)

[SDK for PHP](#)

[SDK for Python](#)

[SDK for Weixin Mini Program](#)

Multipart Copy

Last updated : 2024-03-25 15:28:24

Overview

To copy an object that is greater than 5 GB, you must use multipart copy. First, use the multipart upload API to create an object. Then use the `Upload Part - Copy` API and specify the `x-cos-copy-source` header to determine the source object that you want to copy. The process is outlined below:

1. Initialize an object for multipart upload.
2. Copy the data of the source object; specify the `x-cos-copy-source-range` header to determine how much data to copy at a time (you can specify to copy up to 5 GB at a time).
3. Complete the multipart upload.

Note:

The SDKs provided by COS can be used to easily implement a multipart copy operation.

Directions

Using REST APIs

Use REST APIs to directly initiate a multipart copy request. For more information, see the following API documentation:

[Initiate Multipart Upload](#)

[Upload Part - Copy](#)

[Complete Multipart Upload](#)

[Abort Multipart Upload](#)

Using SDKs

Use multipart copy directly through SDKs. For more information, see the SDK documentation for the corresponding programming language below:

[Android SDK](#)

[C SDK](#)

[C++ SDK](#)

[.NET\(C#\) SDK](#)

[Go SDK](#)

[iOS SDK](#)

[Java SDK](#)

[JavaScript SDK](#)

[Node.js SDK](#)

[PHP SDK](#)

[Python SDK](#)

[Mini Program SDK](#)

Deleting Objects

Deleting a Single Object

Last updated : 2024-03-25 15:28:24

Overview

Cloud Object Storage (COS) supports deleting one or multiple objects directly. To delete one object, you only need to provide its object key and call an API request to delete it.

Directions

Using COS console

Delete an object in the COS console. For more information, see [Deleting an Object](#) in Console Guide.

Using REST API

Use the REST API to initiate a single object deleting request. For more information, see [DELETE Object](#).

Using SDKs

Directly call the single object deleting method in the SDK. For more information, see the SDK documentation for the corresponding programming language below:

[SDK for Android](#)

[SDK for C](#)

[SDK for C++](#)

[SDK for .NET](#)

[SDK for Go](#)

[SDK for iOS](#)

[SDK for Java](#)

[SDK for JavaScript](#)

[SDK for Node.js](#)

[SDK for PHP](#)

[SDK for Python](#)

[SDK for WeChat Mini Program](#)

Deleting Multiple Objects

Last updated : 2024-03-25 15:28:24

Overview

Cloud Object Storage (COS) supports batch deletion of multiple objects. You can delete objects in batches using the console, API, or SDKs.

By default, when the deletion task is completed, an empty response is returned. If an error occurs, an error message is returned.

Note:

A maximum of 1,000 objects can be deleted in a single request. To delete more objects, split the list and send the request separately.

Directions

Using COS console

Delete multiple objects in batches in the COS console. For more information, see [Deleting Objects](#) in Console Guide.

Using REST API

Use the REST API to initiate a multiple objects deleting request. For more information, see [DELETE Multiple Objects](#).

Using SDKs

Directly call the multiple objects deleting method in the SDK. For more information, see the SDK documentation for the corresponding programming language below:

[SDK for Android](#)

[SDK for C](#)

[SDK for C++](#)

[SDK for Go](#)

[SDK for iOS](#)

[SDK for JavaScript](#)

[SDK for Node.js](#)

[SDK for PHP](#)

[SDK for Python](#)

[Mini Program SDK](#)

Data Management

Lifecycle Management

Lifecycle Overview

Last updated : 2024-03-25 15:28:24

Overview

COS supports object-based lifecycle configuration. You can use lifecycle rules to define operations to perform on applicable objects.

Note:

Up to 1,000 lifecycle rules can be added for each bucket.

Use Cases

Log record

With lifecycle configured, logs stored on COS can be automatically archived after 30 days or deleted after 2 years.

Hot/Cold data tiering

Hot data is frequently accessed in a short period of time after the upload, but then is less or not accessed at all after some time. Therefore, you can set the lifecycle rules to store the data uploaded 30 days ago in the STANDARD_IA storage class and data uploaded 60 days ago in ARCHIVE. This process is called data transition.

Archive management

When you use COS for file archive management, you need to save all historical versions of files for a long time according to the compliance requirements in financial, medical, and other industries. In this case, you can configure the lifecycle to transition and store the historical versions of files in the ARCHIVE storage class.

Configuration Items

You need to configure the following elements to create a lifecycle rule:

Objects

The data to be hit during lifecycle execution. You can customize the lifecycle's scope and data types covered within the scope. During lifecycle execution, the specified scope will be scanned, and the operation will be performed on the configured data types within the scope. You can specify the scope according to the following rules:

Specify by prefix: Data can be matched by directory name or filename prefix.

Specify by tag: Data can be filtered by tag.

The following data types can be configured:

Files on the current version: Files on the latest version in the bucket.

Files on historical versions: Objects on historical versions stored after versioning is enabled. For more information on versioning, see [Overview](#).

Delete marker: Marker indicating that the object has been deleted. The lifecycle feature can remove the marker automatically after all historical versions are deleted. For more information on the delete marker, see [Delete Markers](#).

Incomplete multipart uploads: Fragments generated due to incomplete multipart uploads.

Operations

The operation to be performed when the object is hit:

Data transition: transitions objects to STANDARD_IA, INTELLIGENT TIERING, ARCHIVE, or DEEP ARCHIVE after a specified period.

Expiration: Deletes objects after their specified expiration time.

Time

The time condition for triggering the above operations:

Based on number of days: Specify after how many days since an object was last modified that an action will occur.

Notes

Note:

For how to use the lifecycle, see [Configuring Lifecycle](#).

File modification time

Lifecycle supports triggering rule execution based on the object modification time. Only file write operations such as [PUT Object](#), [PUT Object - Copy](#), [POST Object](#), and [Complete Multipart Upload](#) APIs will update the object modification time. The modification time of objects transitioned based on the lifecycle won't be updated.

For example, you configured a lifecycle rule at 15:00 on the 1st day of the month to delete files one day or longer after they are modified. Then, at 00:00 on the 2nd day, the lifecycle task scans for files that were modified over one day ago and deletes them. Files uploaded on the 1st day will not be deleted at 00:00 on the 2nd day, as the time elapsed since their modification is less than one day. Instead, these files will be deleted at 00:00 on the 3rd day.

Note:

A lifecycle can be set to as long as 3,650 days.

Effective time

Data transition

Supported regions

Data transition is supported in public cloud regions. Finance Cloud regions support only data transition to the STANDARD_IA storage class.

One-way transition

Data transition is one-way (STANDARD > STANDARD_IA > ARCHIVE, or STANDARD > ARCHIVE) and cannot be done reversely. You can only call [PUT Object - Copy](#) (for non-ARCHIVE/DEEP ARCHIVE only) or [POST Object restore](#) (for ARCHIVE and DEEP ARCHIVE only) to restore data from a colder storage class to a hotter one.

Eventual consistency

If multiple rules are configured for the same set of objects and conflict with each other (excluding the configuration for deletion upon expiration), COS will execute the rule that **transitions the objects to the coldest storage class**.

For example, if rules A and B are configured to **transition the objects to STANDARD_IA and ARCHIVE** 90 days after file modification respectively and both hit the same object `test.txt`, rule B will be executed.

Rule	Resource	Operation	Time Condition	Execution
Rule A	test.txt	Transition the object to the STANDARD_IA class.	90 days after file modification	Execution fails due to a rule conflict.
Rule B	test.txt	Transition the object to the ARCHIVE class.	90 days after file modification	Execution succeeds.

Note:

It is strongly recommended that you do not configure conflicting lifecycle rules for the same set of objects in COS, because this may result in additional fees.

Transitioning an object won't change the time the object was uploaded or modified.

Deletion upon expiration

How it works

When an object matches the specified lifecycle rule of deletion upon expiration, Tencent Cloud will add the object to the asynchronous deletion queue. There may be a certain delay in actual deletion. You can get the current status of the object by performing GET or HEAD Object operation.

Eventual consistency

If multiple conflicting rules are configured for the same set of objects, the rule that will expire first will be performed first. **Deletion upon expiration will be performed before the transition.**

For example, if rules C and D are configured to **transition the objects to STANDARD_IA and delete the objects** 180 days after file modification respectively and both hit the same object `test.txt`, rule D will be executed.

Rule	Resource	Operation	Time Condition	Execution
Rule C	test.txt	Transition the object to the STANDARD_IA class.	180 days after file modification	Execution fails due to a rule conflict.
Rule D	test.txt	Delete the object.	180 days after file modification	Execution succeeds.

Note:

It is strongly recommended that you do not configure conflicting lifecycle rules for the same set of objects in COS, because this may result in different fees.

Cost considerations

Execution instructions

When the lifecycle feature performs a deletion operation, a backend deletion request will be generated. When it performs a transition operation, backend deletion and write requests will be generated. The requests generated by the above operations will be included in the request bill. For example, transitioning the file `test.txt` in STANDARD storage class to STANDARD_IA through the lifecycle will generate two requests to delete the data in STANDARD and write the data in STANDARD_IA.

When an accident occurs or there are too many objects in the bucket, lifecycle execution may fail. For failures due to other reasons, perform the GET or HEAD Object operation to get the current object status.

Tencent Cloud cannot provide an accurate bill unless the lifecycle execution is completed.

Time-insensitive

Please note that the minimum storage duration is 30 days, 90 days, and 180 days for the STANDARD_IA/INTELLIGENT TIERING, ARCHIVE, and DEEP ARCHIVE storage classes, respectively. No additional storage fees will be incurred for the transition or delete action itself. COS will ignore lifecycle configurations less than 30/90/180 days, and perform only correct configurations upon your request.

For example, if an object in STANDARD_IA is transitioned before 30 days, it will start incurring ARCHIVE storage fees on the transition day and continue incurring STANDARD_IA storage fees until the 30th day. Another example is that if an archived object is deleted upon expiration before it is stored for 90 days, it will continue incurring ARCHIVE storage fees until the 90th day. It works the same way with DEEP ARCHIVE.

Size limits

There is a minimum size limit for objects in the STANDARD_IA, INTELLIGENT TIERING, ARCHIVE, and DEEP ARCHIVE storage classes. For example, if an object smaller than 64 KB is uploaded to the STANDARD_IA storage class, it will be calculated as 64 KB. To reduce user costs, lifecycle execution does not transition the storage classes of objects that are smaller than 64 KB.

Note:

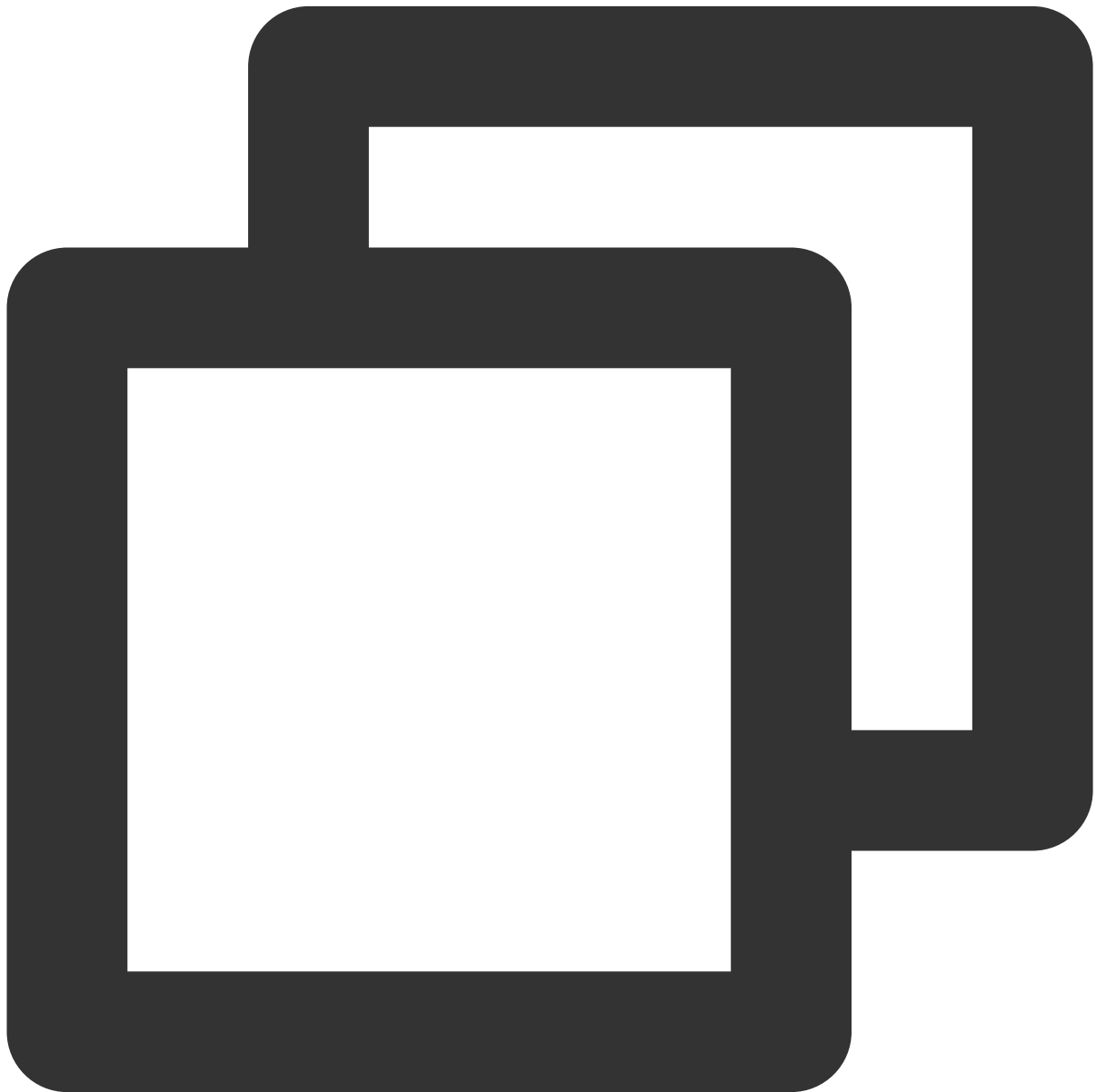
Lifecycle execution does not transition the storage classes of objects that are smaller than 64 KB.

Lifecycle Configuration Elements

Last updated : 2024-03-25 15:28:24

Basic Structure

A lifecycle configuration is specified as XML and consists of one or more lifecycle rules. The structure is as follows:



```
<LifecycleConfiguration>
```

```
<Rule>
  <ID>**your lifecycle name**</ID>
  <Status>Enabled</Status>
  <Filter>
    <And>
      <Prefix>projectA/</Prefix>
      <Tag>
        <Key>key1</Key>
        <Value>value1</Value>
      </Tag>
    </And>
  </Filter>
  **transition/expiration actions**
</Rule>
<Rule>
  ...
</Rule>
</LifecycleConfiguration>
```

Each rule contains the following:

ID (optional): Indicates the content of the rule and is customizable.

Status: Indicates whether the rule is enabled or disabled.

Filter: Identifies objects to which the rule applies.

Action: Specifies actions (operations) that need to be performed on objects that match the above description.

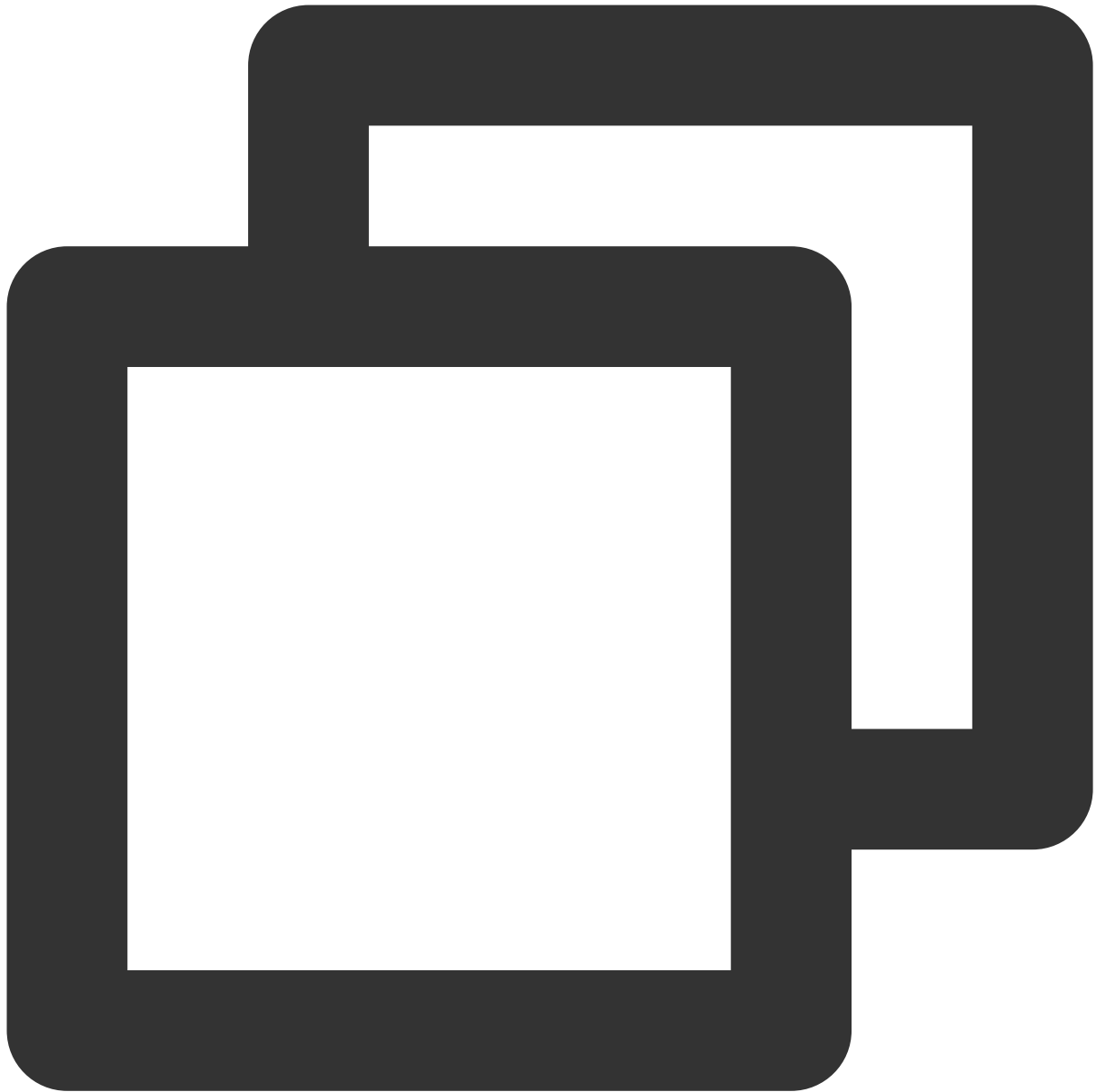
Time: `Days` (after the object's last modified date) or `Date` based on which actions are performed on objects.

Rule Description

Filter element

For all objects in bucket

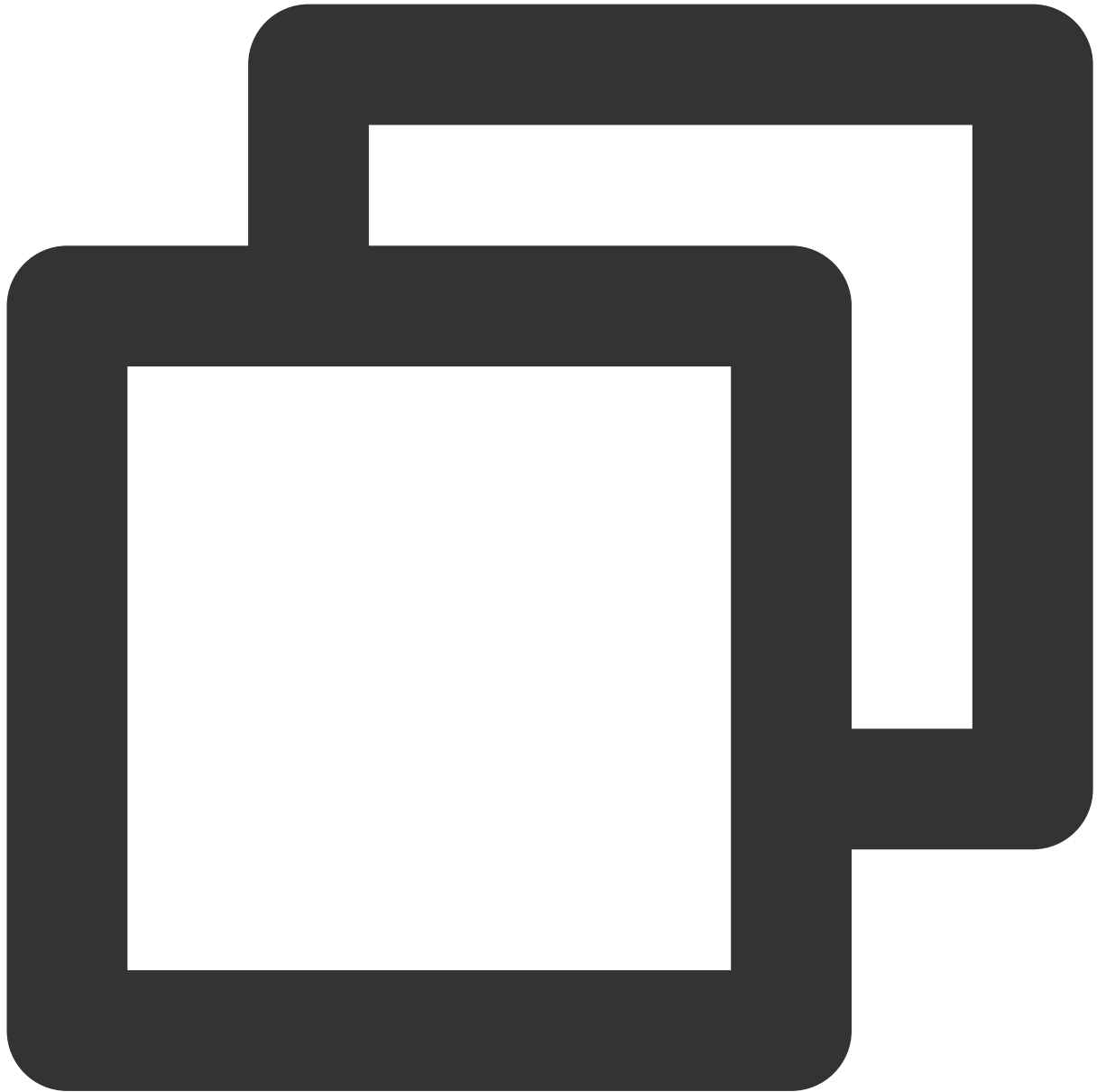
Specify an empty filter, in which case the rule applies to all objects in the bucket.



```
<LifecycleConfiguration>
  <Rule>
    <Filter>
    </Filter>
    <Status>Enabled</Status>
    **transition/expiration actions**
  </Rule>
</LifecycleConfiguration>
```

Based on object key prefix

Specify a prefix-based filter so that the rule applies only to objects with the specified prefix. For example, you can filter out objects by the prefix `logs/` :



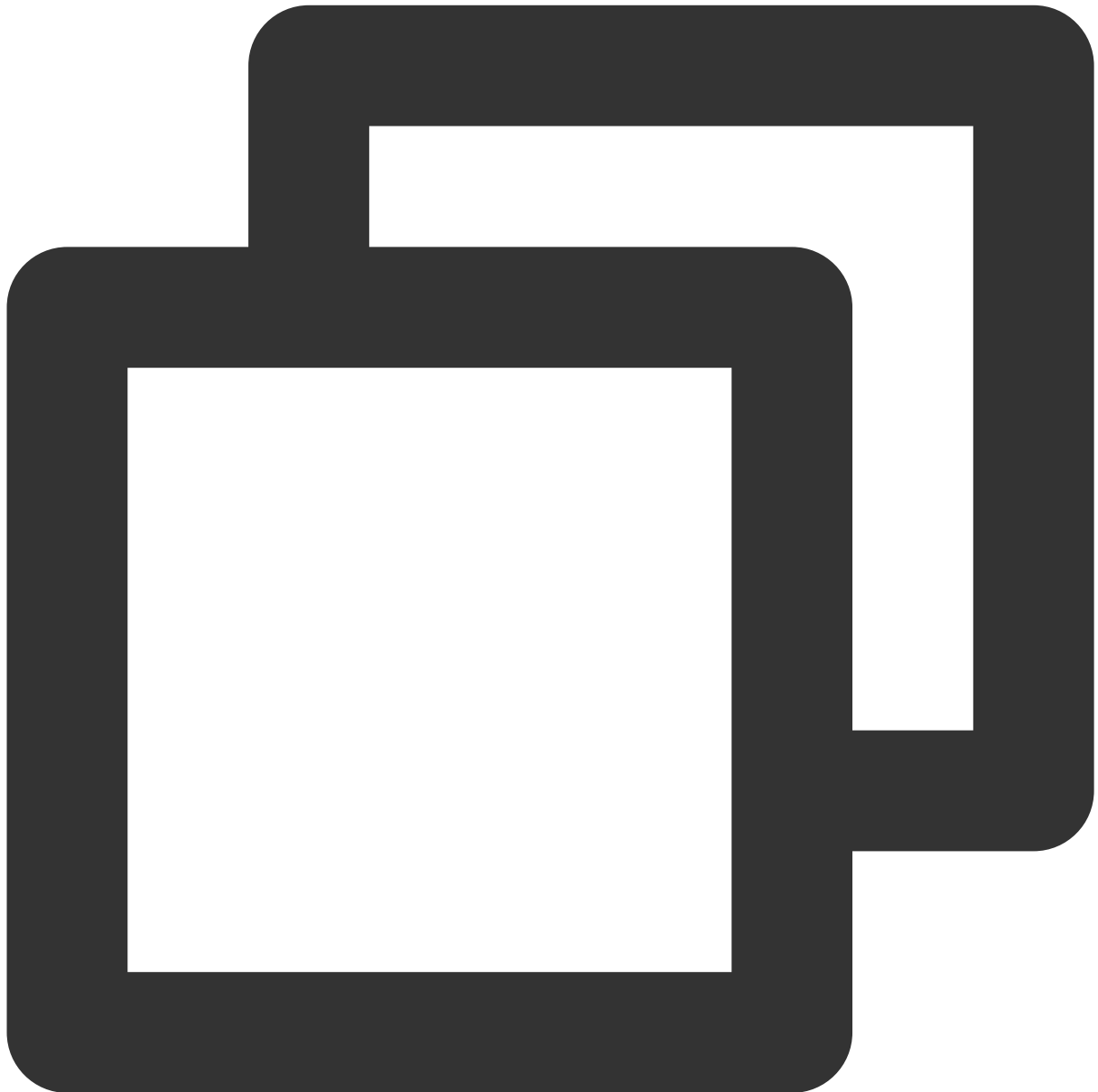
```
<LifecycleConfiguration>
  <Rule>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    **transition/expiration actions**
  </Rule>
```



```
</LifecycleConfiguration>
```

Based on object tag

Specify a filter based on `key` and `value` so that the rule applies only to objects with the specified tag. For example, you can filter out objects by tags `key=type` and `value=image` :



```
<LifecycleConfiguration>
  <Rule>
    <Filter>
      <Tag>
```

```
<Key>type</Key>
<Value>image</Value>
</Tag>
</Filter>
<Status>Enabled</Status>
**transition/expiration actions**
</Rule>
</LifecycleConfiguration>
```

Using multiple filters

In COS, you can combine multiple filters by using `AND` . For example, you can filter out objects by the prefix

`logs/` as well as tags `key=type` and `value=image` :



```
<LifecycleConfiguration>
  <Rule>
    <Filter>
      <And>
        <Prefix>logs/</Prefix>
        <Tag>
          <Key>type</Key>
          <Value>image</Value>
        </Tag>
      </And>
    </Filter>
```

```
<Status>Enabled</Status>
  **transition/expiration actions**
</Rule>
</LifecycleConfiguration>
```

Action element

Specify one or more predefined actions in a lifecycle rule so that these actions will be performed when any objects meet the rule.

Transition action

Specify the `Transition` action to transition objects from one storage class to another. For a versioning-enabled bucket, this action applies to the current object version. You can set the transition date to as short as 0 days. For example, you can transition objects to the ARCHIVE storage class in 30 days:



```
<Transition>
  <StorageClass>ARCHIVE</StorageClass>
  <Days>30</Days>
</Transition>
```

Deletion upon expiration

Specify the `Expiration` action to delete expired objects. For a versioning-disabled bucket, expired objects will be permanently deleted. For a versioning-enabled bucket, expired objects are **added with a `DeleteMarker`** and become the current version. For example, you can delete objects that expire in 30 days:



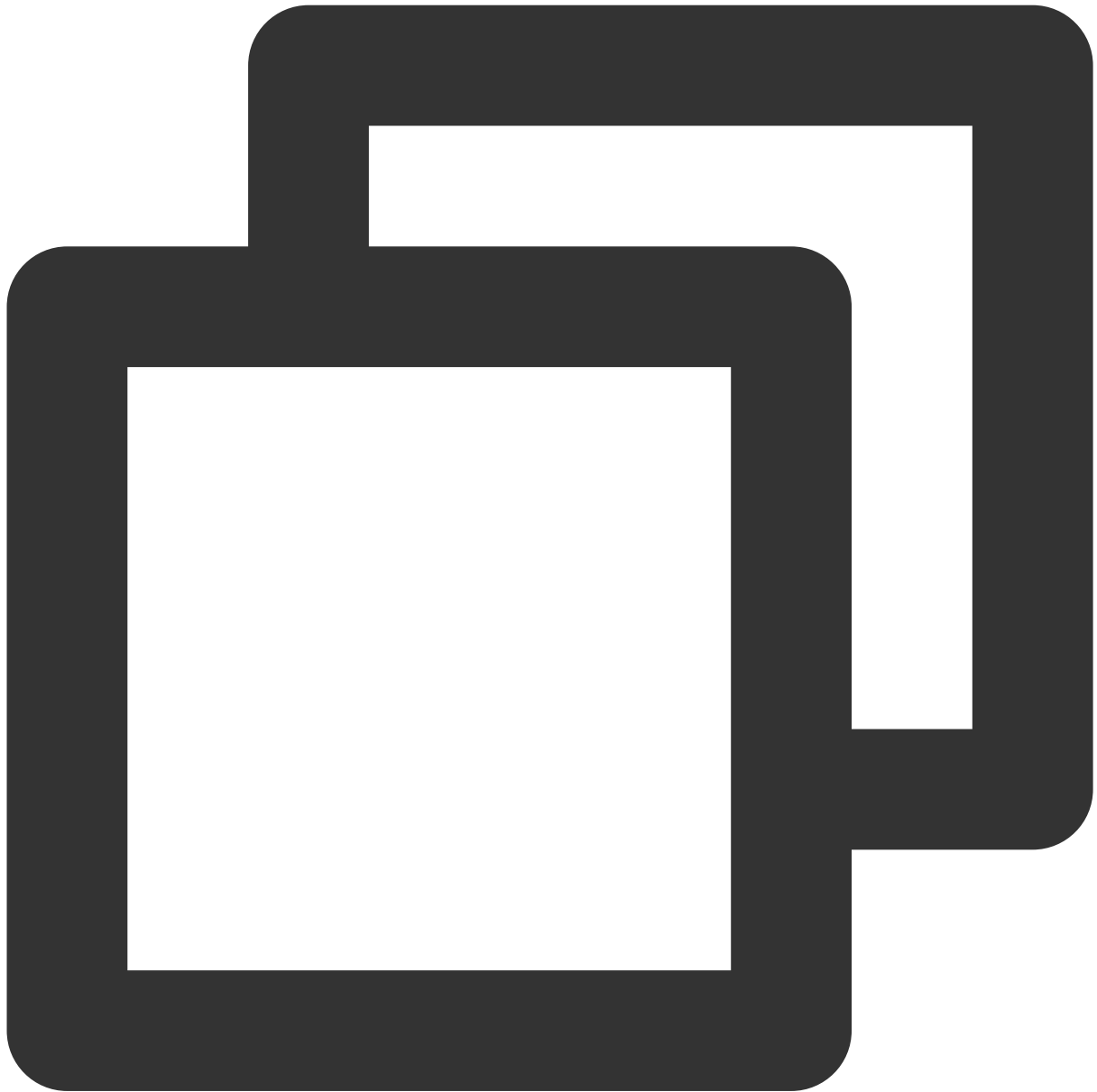
```
<Expiration>  
  <Days>30</Days>  
</Expiration>
```

Incomplete multipart upload

Note:

You cannot specify object tags and clear incomplete multipart uploads at the same time in the same lifecycle rule.

Specify the `AbortIncompleteMultipartUpload` action to delete multipart upload tasks with the specified `UploadId` if they are not successfully completed within the predefined time period. Then, these tasks cannot be resumed or indexed. For example, you can abort multipart upload tasks not successfully completed within 7 days:

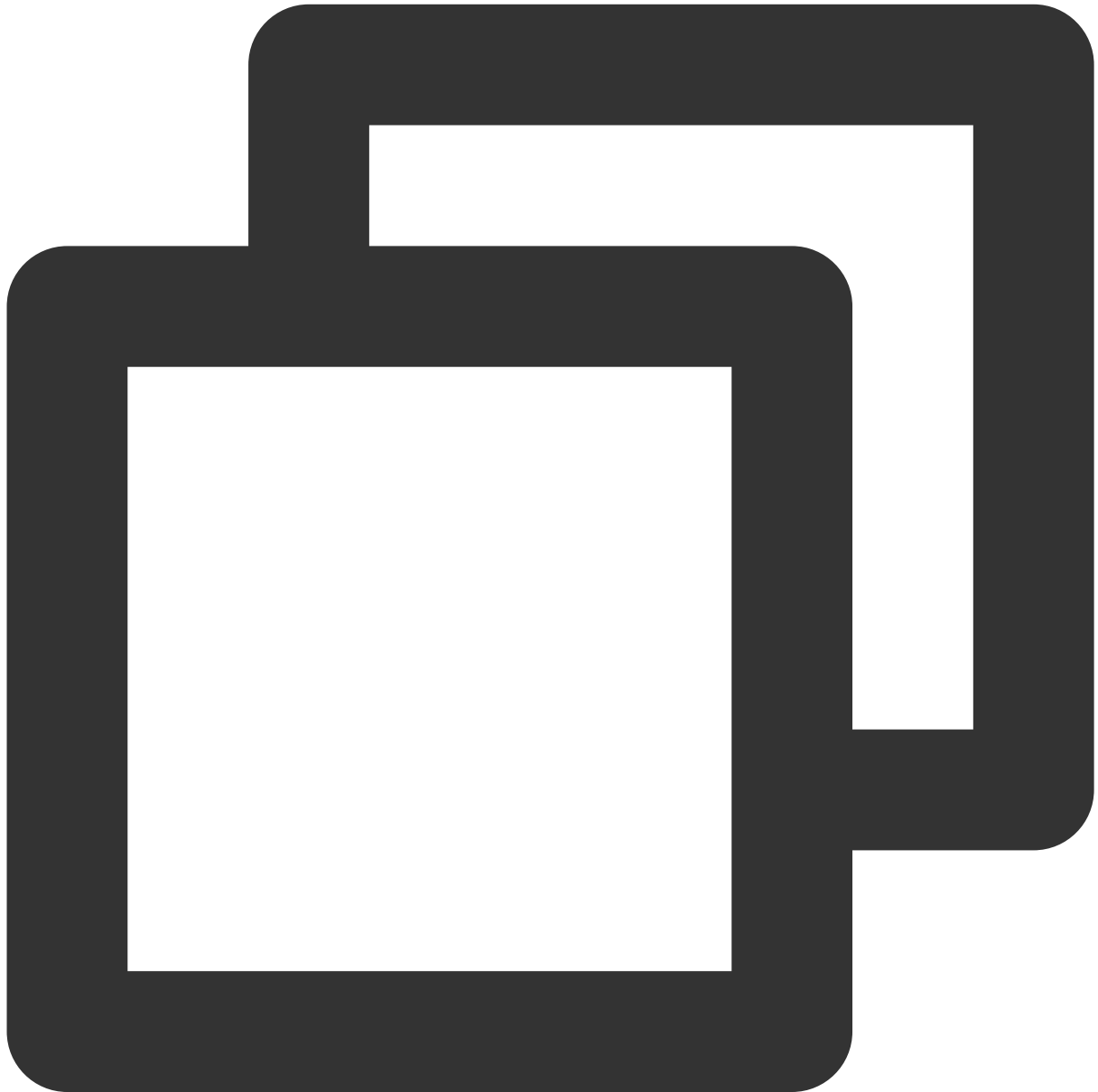


```
<AbortIncompleteMultipartUpload>  
  <DaysAfterInitiation>7</DaysAfterInitiation>  
</AbortIncompleteMultipartUpload>
```

Non-current object

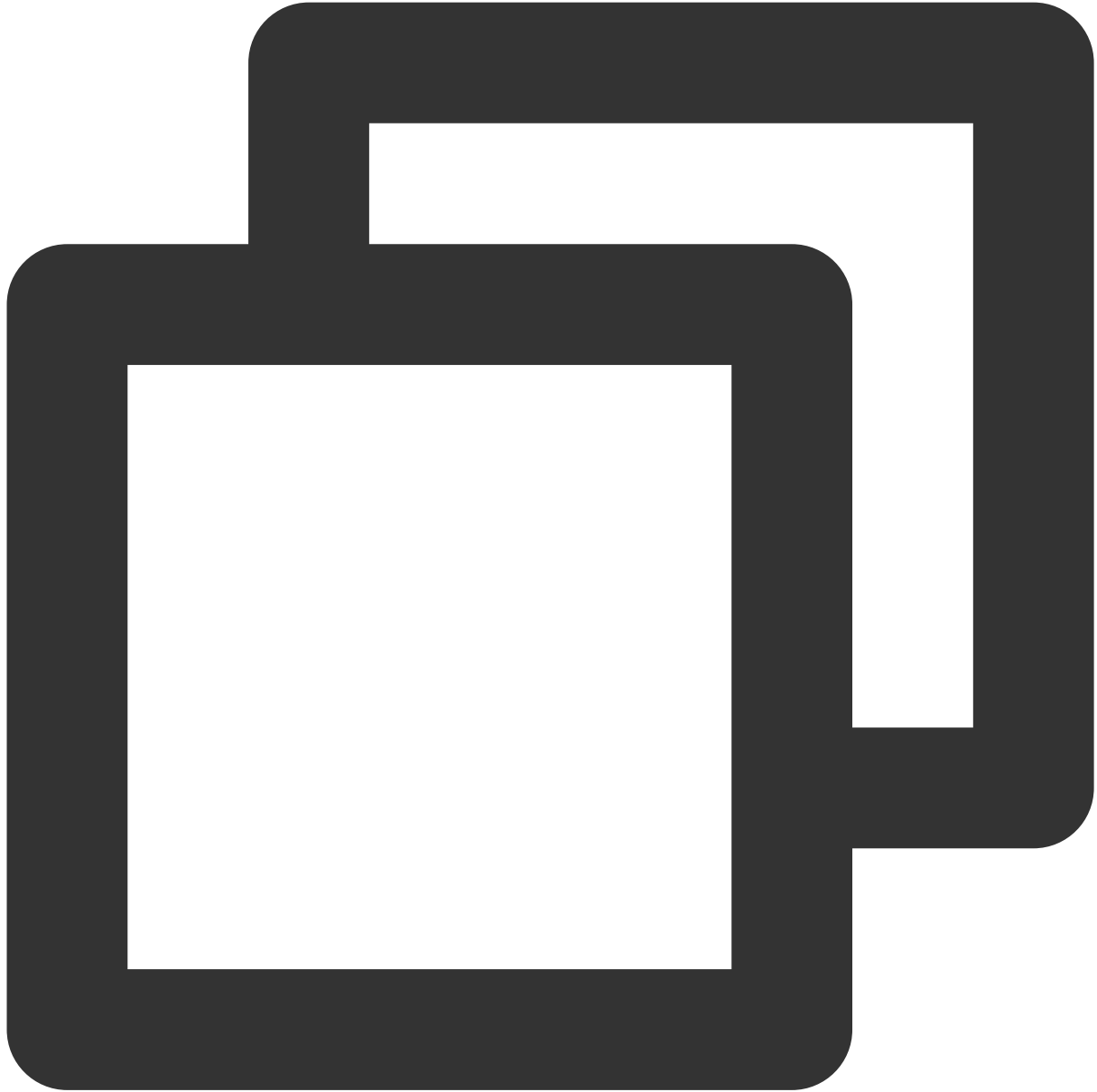
In a versioning-enabled bucket, the `Transition` action only applies to objects of the latest version, and the `Expiration` action only results in adding delete markers to the objects. Therefore, COS provides the following actions for objects of non-current versions:

Specify the `NoncurrentVersionTransition` action to transition non-current objects to another storage class at the specified time. For example, you can transition non-current objects to the `ARCHIVE` storage class in 30 days:



```
<NoncurrentVersionTransition>
  <StorageClass>ARCHIVE</StorageClass>
  <Days>30</Days>
</NoncurrentVersionTransition>
```


Specify the `NoncurrentVersionExpiration` action to delete non-current objects at the specified time. For example, you can delete non-current objects in 30 days:



```
<NoncurrentVersionExpiration>  
  <Days>30</Days>  
</NoncurrentVersionExpiration>
```

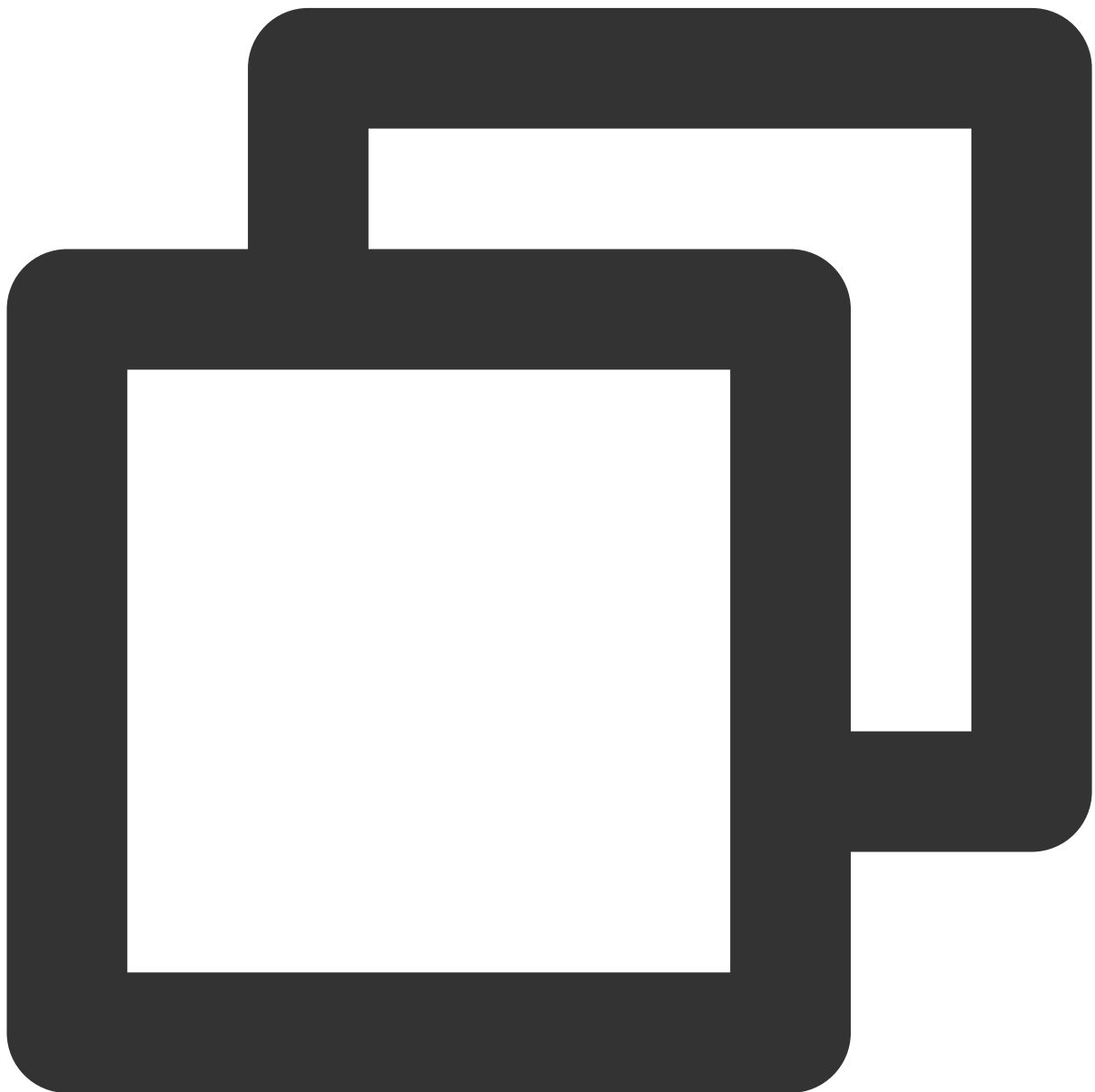
Specify the `ExpiredObjectDeleteMarker` action to clear the remaining delete markers. You can set it to `true` or `false`. This action is triggered by the `NoncurrentVersionExpiration` action. If this feature is

enabled, when lifecycle execution deletes the last historical version in the lifecycle of an object, COS will automatically delete the remaining delete markers.

The details are as follows. If the `NoncurrentVersionExpiration` action is in effect:

If there is only one delete marker left, COS will automatically delete the last delete marker regardless of whether `ExpiredObjectDeleteMarker` is enabled.

If there are multiple delete markers left, COS will check whether `ExpiredObjectDeleteMarker` is enabled (`true`), and if so, COS will automatically delete the remaining delete markers; otherwise, COS will not delete them.



```
<ExpiredObjectDeleteMarker>  
<Days>true|false</Days>
```

```
</ExpiredObjectDeleteMarker>
```

Time element

Based on number of days

`Days` refers to the number of days since an object was last modified.

For example, if an object is set to be transitioned to the `ARCHIVE` storage class in 0 days, when the object is uploaded at 2018-01-01 23:55:00 GMT+8, it will be added to the transition queue at 2018-01-02 00:00:00 GMT+8 and transitioned before 2018-01-02 23:59:59 GMT+8.

For example, if an object is set to be deleted upon expiration in 1 day, when the object is uploaded at 2018-01-01 23:55:00 GMT+8, it will be added to the expiration queue at 2018-01-03 00:00:00 GMT+8 and deleted before 2018-01-03 23:59:59 GMT+8.

Based on specific date

Perform the predefined action on all eligible objects based on the filter on the specified `Date`. The date must be in the ISO 8601 format, and the time is always 00:00 GMT+8.

For example, use `2018-01-01T00:00:00+08:00` for January 1, 2018.

Configuring Lifecycle

Last updated : 2024-03-25 15:28:24

Use Cases

With lifecycle configuration, certain predefined actions can be automatically performed when a rule is applied to objects. For example:

Transition: Transitions objects to the STANDARD_IA, INTELLIGENT_TIERING, ARCHIVE, or DEEP_ARCHIVE storage class after a specified period.

Expiration: Deletes objects after their specified expiration time.

For more information, see [Lifecycle Overview](#) and [Lifecycle Configuration Elements](#).

Directions

Using COS console

Configure the lifecycle in the COS console. For more information, see [Lifecycle Management](#) in Console Guide.

Using REST API

Configure and manage the lifecycle of objects in a bucket through the REST API as described in the following API documentation:

[PUT Bucket lifecycle](#)

[GET Bucket lifecycle](#)

[DELETE Bucket lifecycle](#)

Using SDKs

Directly call the lifecycle method in the SDK. For more information, see the SDK documentation for the corresponding programming language below:

[SDK for Android](#)

[SDK for C](#)

[SDK for C++](#)

[SDK for .NET](#)

[SDK for Go](#)

[SDK for iOS](#)

[SDK for Java](#)

[SDK for JavaScript](#)

[SDK for Node.js](#)

[SDK for PHP](#)

[SDK for Python](#)

[Mini Program SDK](#)

Hosting a Static Website

Last updated : 2024-03-25 15:28:24

Concepts

A static website is a website that contains static content (such as HTML) or client scripts. You can configure static websites for buckets with a custom domain name through the console. A dynamic website contains server scripts such as PHP, JSP, or ASP.NET, and needs to be processed on a server. You can host static websites on Tencent Cloud COS, but cannot write server scripts. For deployment of a dynamic website, we recommend you use a CVM for server code deployment.

Samples

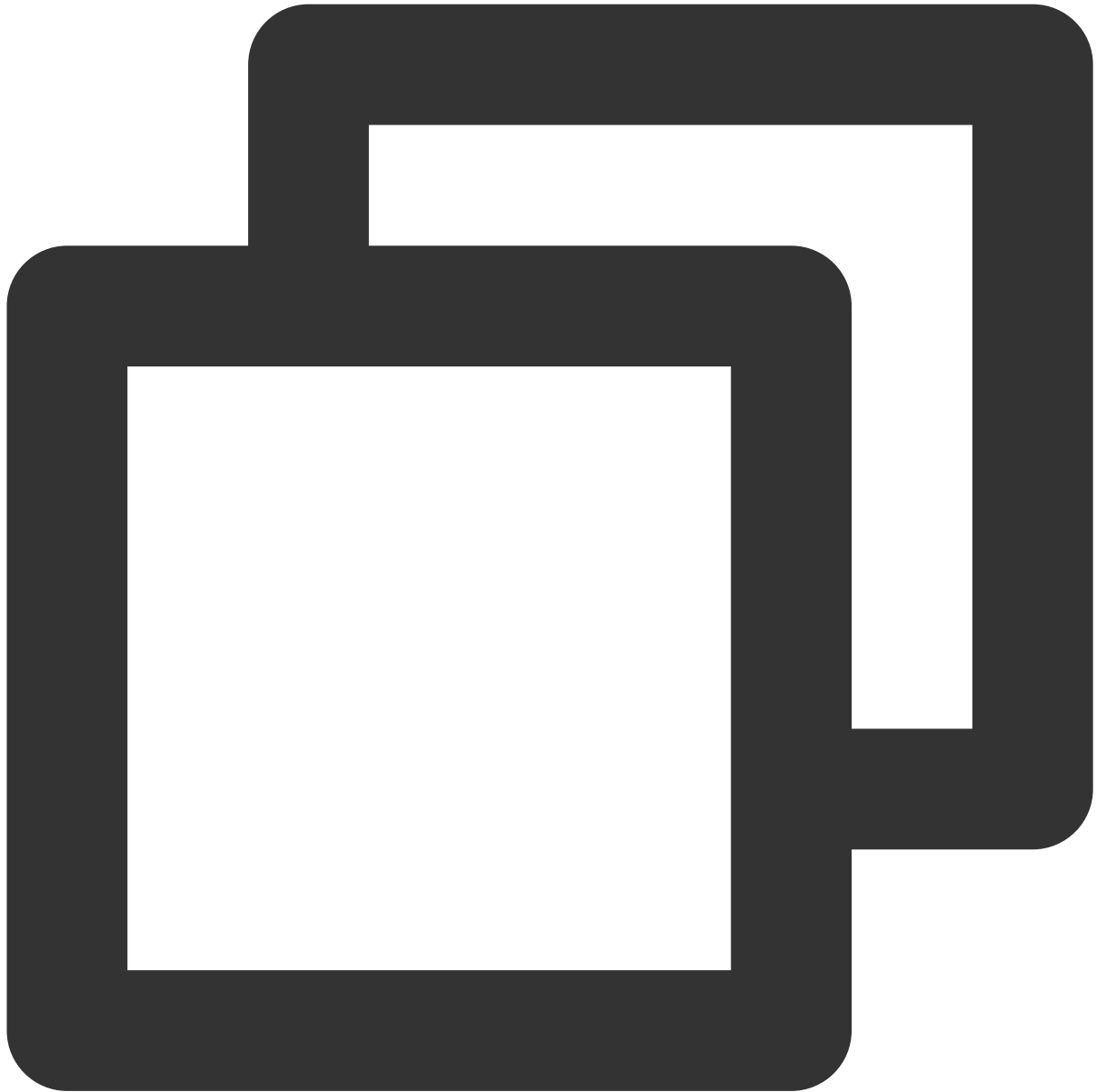
A user created a bucket named examplebucket-1250000000 and uploaded the following files:



```
index.html  
404.html  
403.html  
test.html  
docs/a.html  
images/
```

Static website

Before enabled: Access the bucket via the following default domain name. When a download prompt pops up, save the `index.html` file to the local.



```
https://examplebucket-1250000000.cos-website.ap-guangzhou.myqcloud.com/index.html
```

After enabled: Access the bucket via the following access node, and then you can view the content of `index.html` directly in the browser.



```
https://examplebucket-1250000000.cos-website.ap-guangzhou.myqcloud.com/index.html
```

Forced HTTPS

Before enabled: When the request source is HTTP, the access node URL keeps the HTTP unencrypted transport protocol:



```
http://examplebucket-1250000000.cos-website.ap-guangzhou.myqcloud.com
```

After enabled: The access node always maintains the HTTPS encrypted transport protocol regardless of whether the request source is HTTP or HTTPS:



```
https://examplebucket-1250000000.cos-website.ap-guangzhou.myqcloud.com
```

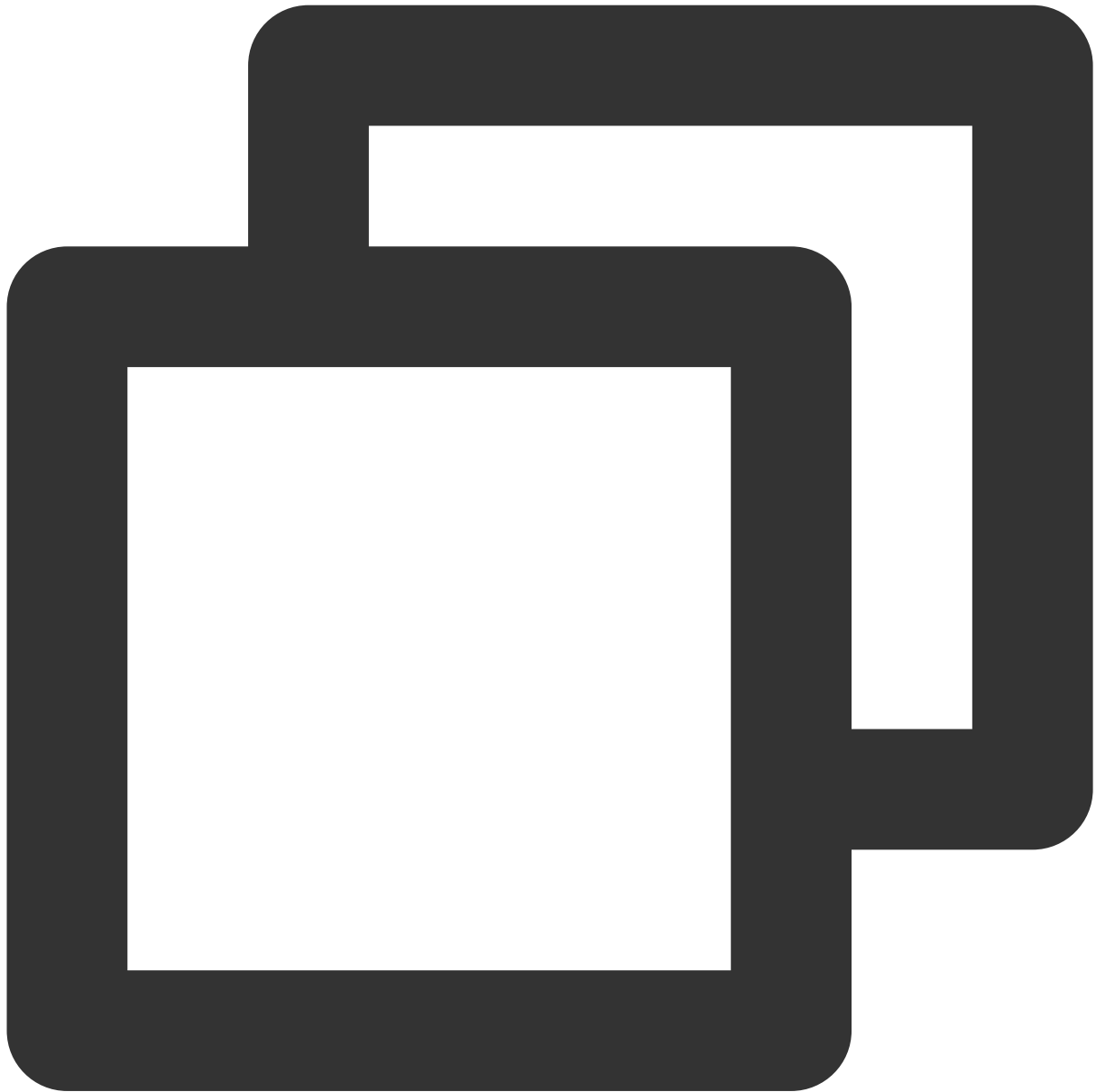
Index document

An index file, the homepage of the static website, is a page returned when the root directory or any subdirectory of a website is requested, and is usually named `index.html`.

When you access a static website via a bucket access domain name, such as

`https://examplebucketbucket-1250000000.cos-website.ap-guangzhou.myqcloud.com`, and no specific page is requested, the web server will return the homepage.

When your user accesses any directory (including the root directory) in a bucket using a URL ending with `/`, the index document in that directory will be matched preferentially. `/` is not mandatory in the root URL, so the index document is returned in response to either of the following URLs.



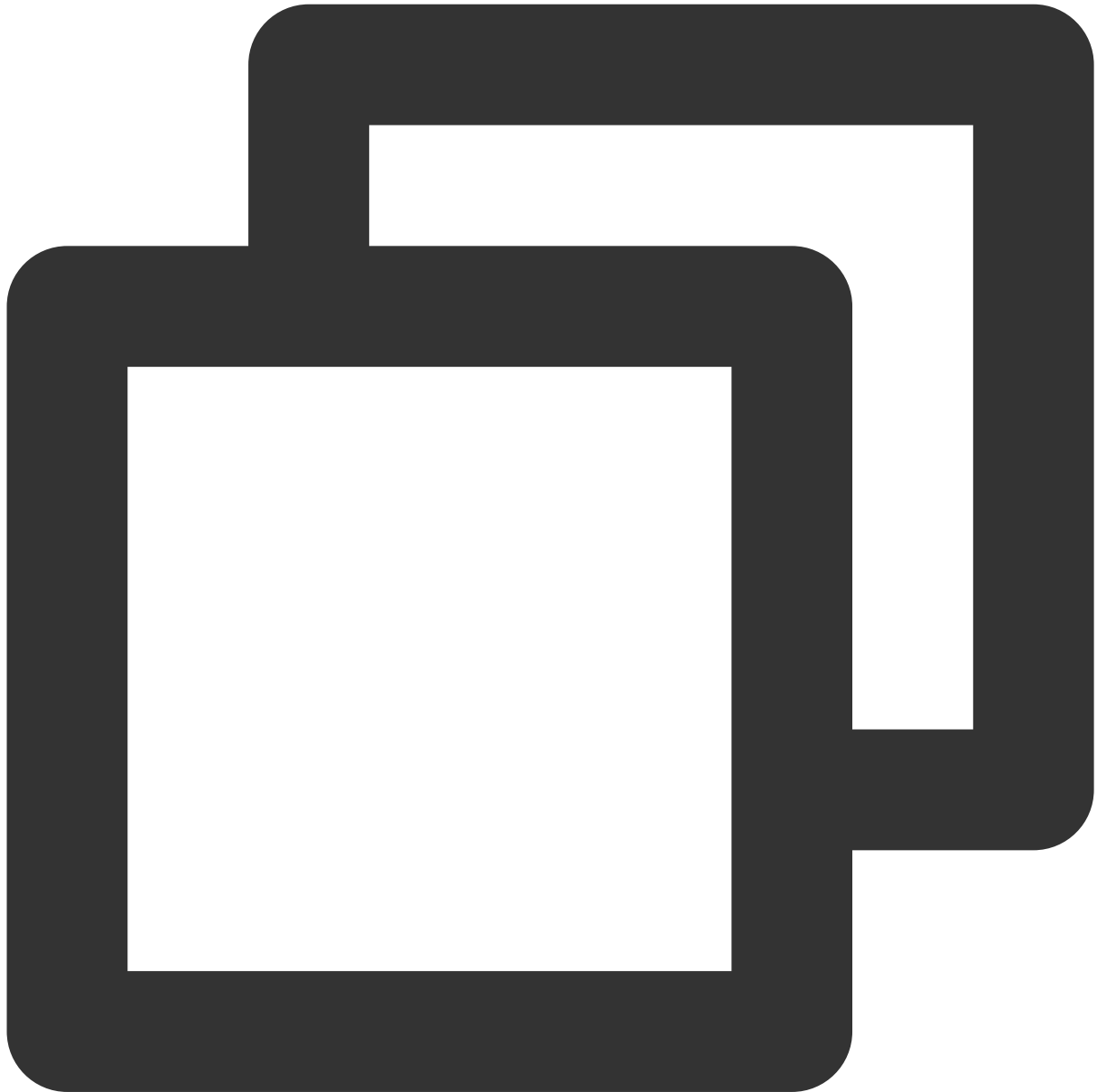
```
http://www.examplebucket.com/  
http://www.examplebucket.com
```

Note:

If a folder is created in the bucket, the index file needs to be added at each level of the folder.

Error document

Assume that when you visit the following page before configuring the error document, a 404 status code is returned, and the default error information is displayed on the page.



```
https://examplebucket-1250000000.cos-website.ap-guangzhou.myqcloud.com/webpage.html
```

When you visit the following page after configuring the error document, a 404 status code is also returned, but the specific error information is displayed on the page.



```
https://examplebucket-1250000000.cos-website.ap-guangzhou.myqcloud.com/webpage.html
```

Redirection rules

Note:

To configure redirection rules for a hosted static website, the path of the replacement file must be an object path in your bucket.

Configure error code redirection

If the webpage.html document is set to **Private Read/Write**, when a user tries to access it, a 403 error is returned. After the 403 error code is redirected to 403.html, the browser will return the content of 403.html. If you do not configure a 403.html document, the browser will return an error document or default error message.

Redirect rules	Type	Description	Force HTTPS	Rule	Replace content	Actions
	Http error code	403	<input checked="" type="checkbox"/>	Replace path	403.html	Delete

Configure prefix match

Note:

Prefix match does not support wildcards. If you want to redirect two folders prefixed with `index1/` and `index2/`, you cannot use `index*/` as the match rule; instead, you should create corresponding match rules separately.

1. When you rename a folder from `docs/` to `documents/`, the user will get an error when accessing the `docs/` folder. So, you can redirect the request with the prefix `docs/` to `documents/`.

Redirect rules	Type	Description	Force HTTPS	Rule	Replace content	Actions
	Prefix matching	docs/	<input checked="" type="checkbox"/>	Replace prefix	documents/	Delete
Add Rules						

2. When you delete the `images/` folder (i.e., deleting all objects with the prefix `images/`), you can add a redirection rule to redirect requests for any object with the prefix `images/` to `test.html`.

Redirect rules	Type	Description	Force HTTPS	Rule	Replace content	Actions
	Prefix matching	images/	<input checked="" type="checkbox"/>	Replace prefix	test.html	Delete
Add Rules						

Inventory Overview

Last updated : 2024-03-25 15:28:24

What Is Inventory?

The inventory feature helps users manage objects in a bucket. It operates periodically as an alternative to the COS synchronous List API. COS can daily or weekly scan a specified object or objects with the same prefix in a bucket based on an inventory job configured, and export and save the inventory report in CSV format to a specified bucket.

The report lists the objects stored and their metadata, and records the desired object attributes as configured.

You can use the inventory feature for various purposes, including but not limited to:

Reviewing and reporting the replication and encryption status of objects.

Simplifying and accelerating business workflows and big data jobs.

Note:

You can configure multiple inventory jobs in one bucket. Such jobs do not directly read the object content during their execution; instead, they only scan the attribute information such as object metadata.

Inventory Parameters

After you configure an inventory job for a bucket, COS regularly scans specified objects in the bucket based on your configuration, and exports an inventory report in .csv format. Currently, the inventory report can record the following information:

Inventory Information	Description
AppID	Account ID
Bucket	Name of the bucket where the inventory job is performed
fileFormat	File format
listObjectCount	Number of objects to list, which will be used for billing. For more information, see the inventory feature fees in Management Feature Fees .
listStorageSize	Size of listed objects
filterObjectCount	Number of filtered objects
filterStorageSize	Size of filtered objects
Key	Name of an object file in a bucket. When the CSV file format is used, the key name is

	URL-encoded and must be decoded before you can use it.
VersionId	Version ID of an object. After versioning is enabled for a bucket, COS specifies a version ID for the object added to the bucket. If the inventory is for the current version only, this field is not included.
IsLatest	Set to True if the latest object version is used. This field is not included if the inventory is only for the current version of objects.
IsDeleteMarker	Set to True if the object is a delete marker. This field is not included if the inventory is only for the current version of objects.
Size	Object size in bytes
LastModifiedDate	The last modified date of the object
ETag	Hash value of an object. It displays only modification to the content of an object, rather than to its metadata. The ETag may be or may not be the MD5 checksum of the data of the object, and this depends on how the object is created and encrypted.
StorageClass	Storage class of the object. For more information, see Storage Class
IsMultipartUploaded	Set to True if the object is uploaded in multiple parts. For more information, see Multipart Upload
Replicationstatus	Status of source and replica files in object replication. Source file status: <code>PENDING</code> (to be replicated), <code>COMPLETED</code> (replicated), or <code>FAILED</code> (failed). Replica file status: <code>REPLICA</code> (replicated with replica file generated). For more information, see Actions .
Tag	Object tag

How to Configure Inventory

Before configuring an inventory, you need to understand two concepts:

Source bucket: The bucket for which you want to enable inventory, containing:

Objects listed in the inventory

Configuration of the inventory

Destination bucket: The bucket where to store the inventory report, containing:

An inventory list file

Manifest files that describe the location of the inventory list file

An inventory can be configured as follows:

Specify the information of the objects to be analyzed in the source bucket

To inform COS of object information to be analyzed, you need to configure the following information in the source bucket for the inventory:

Select object versions: List all the versions of the object or list only the current version. If you select to list all the versions of the object, COS adds information about the object in all historical versions to the inventory report. If you select to list only the current version, COS records only the object in the latest version.

Configure the attributes of the objects to be analyzed: You need to tell COS which information in the object attributes needs to be included in the inventory report. Currently, supported object attributes include account ID, source bucket name, object file name, object version ID, whether the object is of the latest version, whether the object is a deletion flag, object size, object's last modified date, ETag, object's storage class, cross-region replication tag, and whether the object is a part in multipart upload.

Configure the storage information for the inventory report

You need to tell COS how often to export the inventory report, which bucket to store the report in, and whether the report should be encrypted. The configuration information is as follows:

Select an export frequency: Daily or weekly. COS will execute the inventory feature at the specified frequency.

Select an encryption mode: No encryption or SSE-COS. If SSE-COS is selected, COS encrypts the generated inventory report.

Configure the output location of the inventory: You need to specify the bucket where to store the inventory report.

Note:

The destination bucket must be in the same region as the source bucket. They can be the same bucket.

How to Use

Configuring inventory via the console

Refer to [Enabling Inventory](#) to learn how to configure the inventory feature in the console.

Configuring inventory via APIs

To enable the inventory feature for a specified bucket using APIs, follow the steps below:

1. Create a COS role.
2. Bind permissions to the COS role.
3. Enable inventory.

1. Creating a COS role

Create a COS role. For more information about the API, see [CreateRole](#).

Here, `roleName` must be `COS_QcsRole`.

`policyDocument` must be:



```
{
  "version": "2.0",
  "statement": [{
    "action": "name/sts:AssumeRole",
    "effect": "allow",
    "principal": {
      "service": "cos.cloud.tencent.com"
    }
  }]
}
```

2. Binding permissions to the COS role

Grant the log role permissions. For more information about the API, see [AttachRolePolicy](#).

Here, `policyName` is set to `QcloudCOSFullAccess`, `roleName` is the `COS_QcsRole` in step 1, or the `roleID` returned when `roleName` was created.

3. Enabling inventory

Call the API to enable inventory. For more information about the API, see [PUT Bucket inventory](#). Note that the destination bucket to store the inventory files should be in the same region as the source bucket.

Storage Path of the Inventory Report

The inventory report and manifest files are put in the destination bucket. The inventory report is located in the following path:



```
destination-prefix/appid/source-bucket/config-ID/
```

Manifest files are located in the following paths in the destination bucket:



```
destination-prefix/appid/source-bucket/config-ID/YYYYMMDD/manifest.json  
destination-prefix/appid/source-bucket/config-ID/YYYYMMDD/manifest.checksum
```

The meanings of the paths are as follows:

destination-prefix: This is the "destination prefix" set when you configure the inventory, which can be used to group all inventory reports in a public location in the destination bucket.

source-bucket: This is the name of the source bucket corresponding to the inventory report. This folder is added to avoid conflicts that may occur when multiple source buckets send their inventory reports to the same destination bucket.

config-ID: This is the "inventory name" set when you configure the inventory. If multiple inventory reports are configured in the same source bucket and delivered to the same destination bucket, config-ID can be used to distinguish among different reports.

YYYYMMDD: Timestamp, including the time and date when the bucket scan is started when the inventory report is generated.

manifest.json: This is the manifest file.

manifest.checksum: This is the MD5 of the content of the manifest.json file.

There are two manifest files: manifest.json and manifest.checksum.

Note:

The following describes the manifest files:

Both manifest.json and manifest.checksum are manifest files. manifest.json describes the location of the inventory report, and manifest.checksum is the MD5 of the content of the manifest.json file. Each time a new inventory report is delivered, it is accompanied by a new set of manifest files.

Each manifest contained in manifest.json provides metadata and basic information about the inventory, including:

Source bucket name.

Destination bucket name.

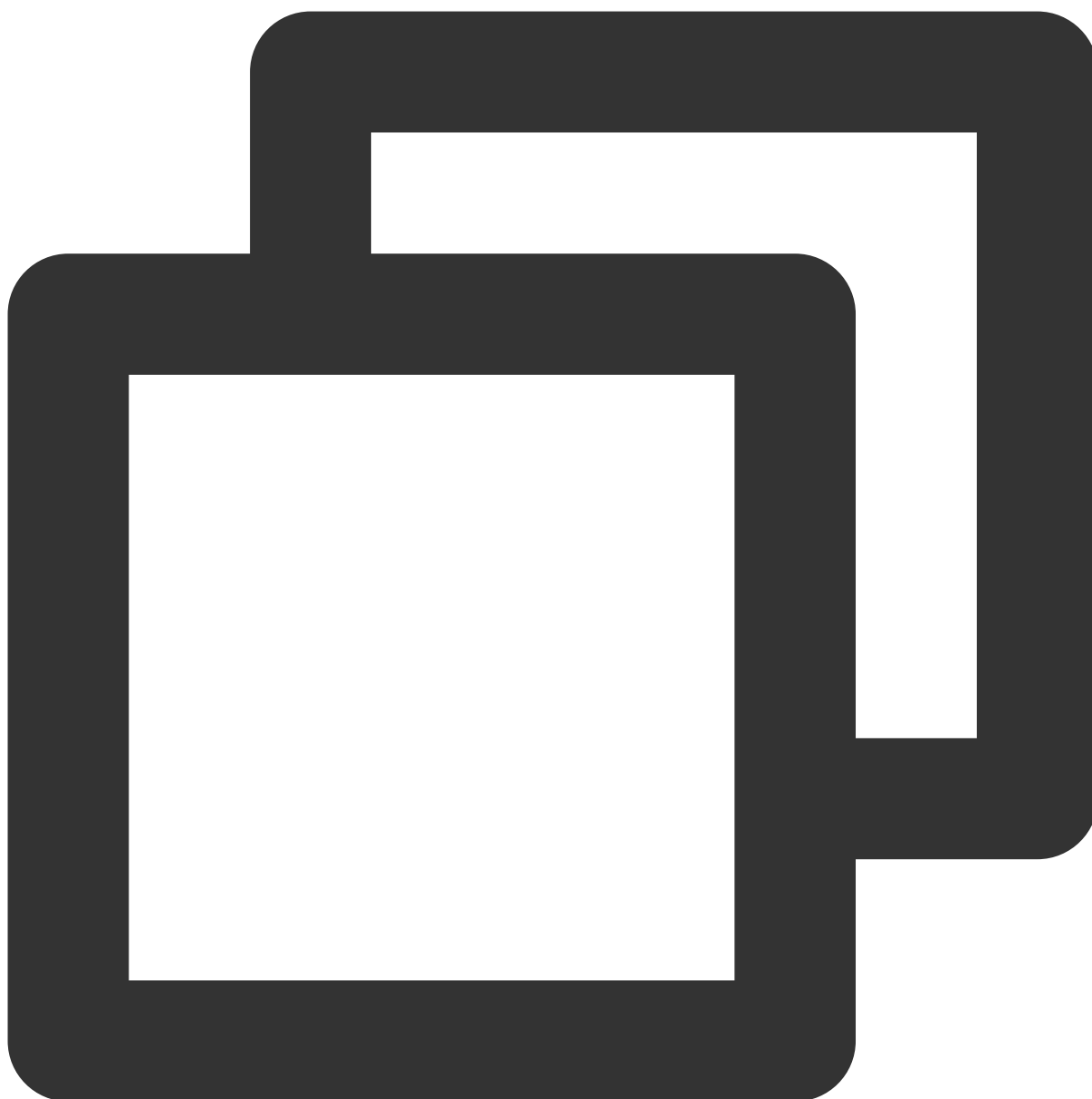
Inventory version.

Timestamp, including the time and date when the bucket scan is started when the inventory report is generated.

Format and architecture of the inventory file.

Object key, size, and md5Checksum of the inventory report in the destination bucket.

The following example shows a manifest in the manifest.json file for CSV-formatted inventory:



```
{
  "sourceAppid": "1250000000",
  "sourceBucket": "example-source-bucket",
  "destinationAppid": "1250000000",
  "destinationBucket": "example-inventory-destination-bucket",
  "fileFormat": "CSV",
  "listObjectCount": "13",
  "listStorageSize": "7212835",
  "filterObjectCount": "13",
  "filterStorageSize": "7212835",
  "fileSchema": "Appid, Bucket, Key, Size, LastModifiedDate, ETag, StorageClass, IsM
```



```
"files": [  
  {  
    "key": "cos_bucket_inventory/1250000000/examplebucket/inventory01/04d73d9debc73d",  
    "size": "502",  
    "md5Checksum": "7d40288a09c25b302ad6cb5fced54f35"  
  }  
]  
}
```

Inventory consistency

All of your objects might not appear in each inventory list. The inventory list provides eventual consistency for PUTs of both new objects and overwrites, and DELETES. Therefore, the inventory report possibly does not include the latest added or deleted object. For example, if the user uploads or deletes an object when COS is executing an inventory job configured by the user, the results of the upload or deletion operations may not be reflected in the inventory report.

If you want to verify the object status before the execution, we recommend you call the [HEAD Object](#) API to extract the object metadata, or check the object attributes in the COS console.

Bucket Tag Overview

Last updated : 2024-03-25 15:28:24

Overview

A bucket tag is a key-value pair (key = value), which is comprised of a tag key and a tag value that are connected by an equal sign ("="), for example: group = IT. It can be used to manage buckets in groups. You can set, query, and delete the tag for a specified bucket.

Specifications and Limits

Limits on Tag Keys

A tag key starting with "qcs:" or "project" is a default tag key and cannot be created.

A tag key can only contain characters encoded in UTF-8, spaces, numbers, or special characters including `+ - = . _ : / @`.

A tag key should be a combination of 0-127 characters encoded in UTF-8.

Tag keys are case sensitive.

Limits on Tag Values

A tag value can only contain characters encoded in UTF-8, spaces, numbers, or special characters including `+ - = . _ : / @`.

A tag value should be a combination of 0-255 characters encoded in UTF-8.

Tag values are case sensitive.

Limits on the Number of Tags

Bucket dimension: A resource has at most 50 different bucket tags.

Tag dimension:

A user has at most 1,000 different keys.

A key has at most 1,000 values.

Multiple same keys are not allowed in the same bucket.

Usage

You can set bucket tags using the console or APIs.

Via the COS Console

For more information about setting a bucket tag using the COS console, see [Setting Bucket Tags](#) in Console Guide.

Via REST APIs

Manage bucket tags using the following APIs:

[PUT Bucket tagging](#)

[GET Bucket tagging](#)

[DELETE Bucket tagging](#)

Object Tag Overview

Last updated : 2024-03-25 15:28:24

Overview

Object tagging is designed to help you group and manage objects in your bucket by adding a key-value pair as an object tag. An object tag consists of a `tagKey` , a `=` , and a `tagValue` , such as `group = IT` . You can set, query, and delete tags on the specified object.

Note:

Object tagging is a paid feature. For detailed pricing, see [Pricing | Cloud Object Storage](#).

Directions

Through the COS console

You can manage object tags in the COS console as instructed in [Setting Object Tag](#).

Through RESTful APIs

You can manage object tags through the following APIs:

[PUT Object tagging](#)

[GET Object tagging](#)

[DELETE Object tagging](#)

Specifications and Restrictions

Tag key restrictions

A tag key can only contain UTF-8 letters, spaces, digits, or special symbols `+ - = . _ : / @ .`

A tag key can contain 1–127 UTF-8 characters.

Tag keys are case-sensitive.

Tag value restrictions

A tag value can only contain UTF-8 letters, spaces, digits, or special symbols `+ - = . _ : / @ .`

A tag value can contain 1–255 UTF-8 characters.

Tag values are case-sensitive.

Tag quantity restrictions

Object dimension: Up to ten unique tags per object.

Tag dimension: Unlimited.

Event Notifications

Last updated : 2024-03-25 15:28:24

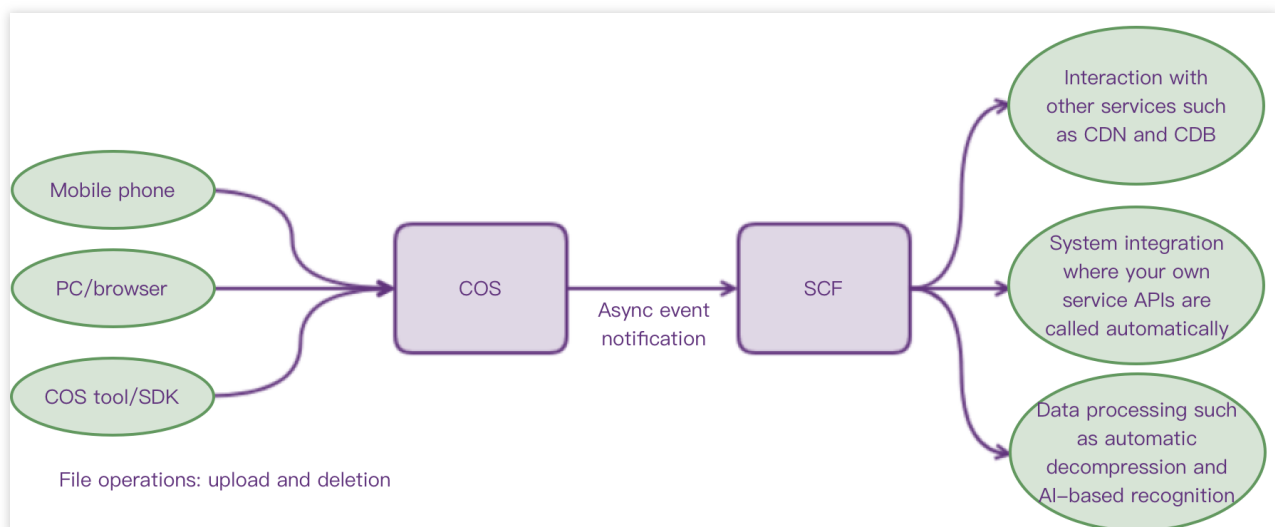
Overview

If any change is made to Cloud Object Storage (COS) resources (such as new files uploaded or files deleted), you will receive prompt notification. Event notification can be used together with [Serverless Cloud Function \(SCF\)](#) to meet the needs of more application scenarios:

Interaction with other Tencent Cloud services: For example, purge CDN cache or update the database when a new file is uploaded to COS.

System integration: Your own service APIs can be called when files in COS are created, deleted, or overwritten. In user-generated content (UGC) scenarios, with the event notification feature, the client side will be able to interact with the server side.

Data processing: Files in COS can be automatically processed, such as automatic decompression and AI recognition.



COS event notification has the following features:

Async processing: sending notifications does not affect normal COS operations.

Notification targets: notifications can only be sent to SCF functions in the same region.

Currently, an SCF function can be triggered by the following COS events:

Event Type	Description
cos: ObjectCreated:*	All upload events mentioned below

cos: ObjectCreated:Put	An object is created using the Put Object API
cos: ObjectCreated:Post	An object is created using the Post Object API
cos: ObjectCreated:Copy	An object is created using the Put Object - Copy API
cos: ObjectCreated:CompleteMultipartUpload	An object is created using the Complete Multipart Upload API
cos:ObjectCreated:Origin	Image origin-pull occurs
cos:ObjectCreated:Replication	An object is created using cross-region replication
cos: ObjectRemove:*	All deletion events mentioned below
cos: ObjectRemove>Delete	An object is deleted from an unversioned bucket using the Delete Object API, or an object with a specified version is deleted by specifying versionid
cos: ObjectRemove>DeleteMarkerCreated	An object is deleted from a versioning-enabled or versioning-suspended bucket using the Delete Object API
cos:ObjectRestore:Post	An ARCHIVE restoration job is created
cos:ObjectRestore:Completed	An ARCHIVE restoration job is completed

How to Use COS Event Notification

Please follow the steps below:

1. Create an SCF function

You can create a function in the [SCF console](#) or using the CLI. When creating the function, you need to select the runtime environment based on the language you will use to write the function and submit the function code either by editing online or uploading a local code package.

You can also use a preconfigured SCF template to simplify the creation. For more information, see [Create a Function](#).

The way to write the function varies by programming language. For more information, see the [SCF](#) documentation.

2. Test the function

Once a function is created, you can test it with a test template which can simulate COS events and trigger the function. For more information, see [Testing a Function](#).

3. Add a trigger

After you finish the testing, you can bind the SCF function with a bucket by creating a COS trigger in the console or

using the command line. For more information, see [Creating a Trigger](#).

4. Check if the function works

After completing the steps above, you can make changes to the bucket in COS to see if everything works. For example, you can upload or delete files via the console or COSBrowser. Then, go to the [SCF console](#), choose **Function Service**, click the name of the target function, and click **Log Query** to check if everything works properly. For more information on SCF COS triggers, see [COS Trigger](#).

Data Extraction

SELECT Overview

Last updated : 2024-03-25 15:28:24

The COS Select feature uses Structured Query Language (SQL) statements to filter the objects stored in COS so as to extract specific objects and get desired data. With COS Select, you can reduce the amount of data transferred by COS for lower costs and latency during data extraction.

The COS Select feature currently allows you to extract objects stored in CSV, JSON, and Parquet formats and compressed by gzip or bzip2 (for CSV and JSON objects only). In addition, you can save extraction results in CSV and JSON formats and specify how to separate the result records.

You can pass in a SQL expression to COS in your request. COS Select currently only supports certain SQL expressions. For more information, see [SQL Functions](#).

You can use the COS console, API, SDK, or COSCMD to perform SQL queries. Note that certain limits apply to file extraction in the COS console: up to 128 MB of files can be extracted, and up to 40 MB of data can be returned. To extract more data, you need to use other methods.

Note:

For more information on data types supported by COS Select and current reserved fields, see [Data Types](#) and [Reserved Fields](#).

-Currently, the extraction function only supports public cloud regions in the Chinese mainland.

Restrictions

The following restrictions apply to COS Select:

You must have the `cos:GetObject` permission to the queried object. A root account has this permission by default.

Only objects in the STANDARD storage class can be extracted.

The maximum length of a SQL expression is 256 KB.

The maximum length of a single record in the extraction result is 1 MB.

SQL clauses currently supported by COS Select include:

SELECT statement

FROM clause

WHERE clause

LIMIT clause

Note:

For more information on SQL clauses, see [SELECT Command](#).

Functions currently supported by COS Select include:

Aggregate functions, such as AVG, COUNT, MAX, MIN, and SUM.

Condition functions, such as COALESCE and NULLIF.

Conversion functions, such as CAST.

Date functions, such as DATE_ADD, DATE_DIFF, EXTRACT, TO_STRING, TO_TIMESTAMP, and UTCNOW.

String functions, such as CHAR_LENGTH, CHARACTER_LENGTH, LOWER, SUBSTRING, TRIM, and UPPER.

Note:

For more information on SQL functions, see [SQL Functions](#).

COS Select currently supports the following operators:

Logical operators: AND, NOT, OR

Comparison operators: <, >, <=, >=, =, <>, !=, BETWEEN, IN

Pattern matching operators: LIKE

Mathematical operators: +, -, *, %

Note:

For more information on operators, see [Operators](#).

Initiating Extraction Request

You can initiate an extraction request using the console, API, or SDK:

If you use the console, see [Data Extraction](#).

If you use the API, see [SELECT Object Content](#).

If you use the SDK, go to [SDK Overview](#) and select the required SDK API.

FAQs

If a problem occurs when you perform a query, COS Select will return an error code and error message. For the list of error codes and descriptions, see [Special Error Codes](#).

SELECT Command

Last updated : 2024-03-25 15:28:24

Overview

The COS Select feature only supports the SELECT command to extract the required data and reduce the amount of data transferred, which helps lower the costs and request delay. The following are the standard clauses supported by SELECT queries:

SELECT statement

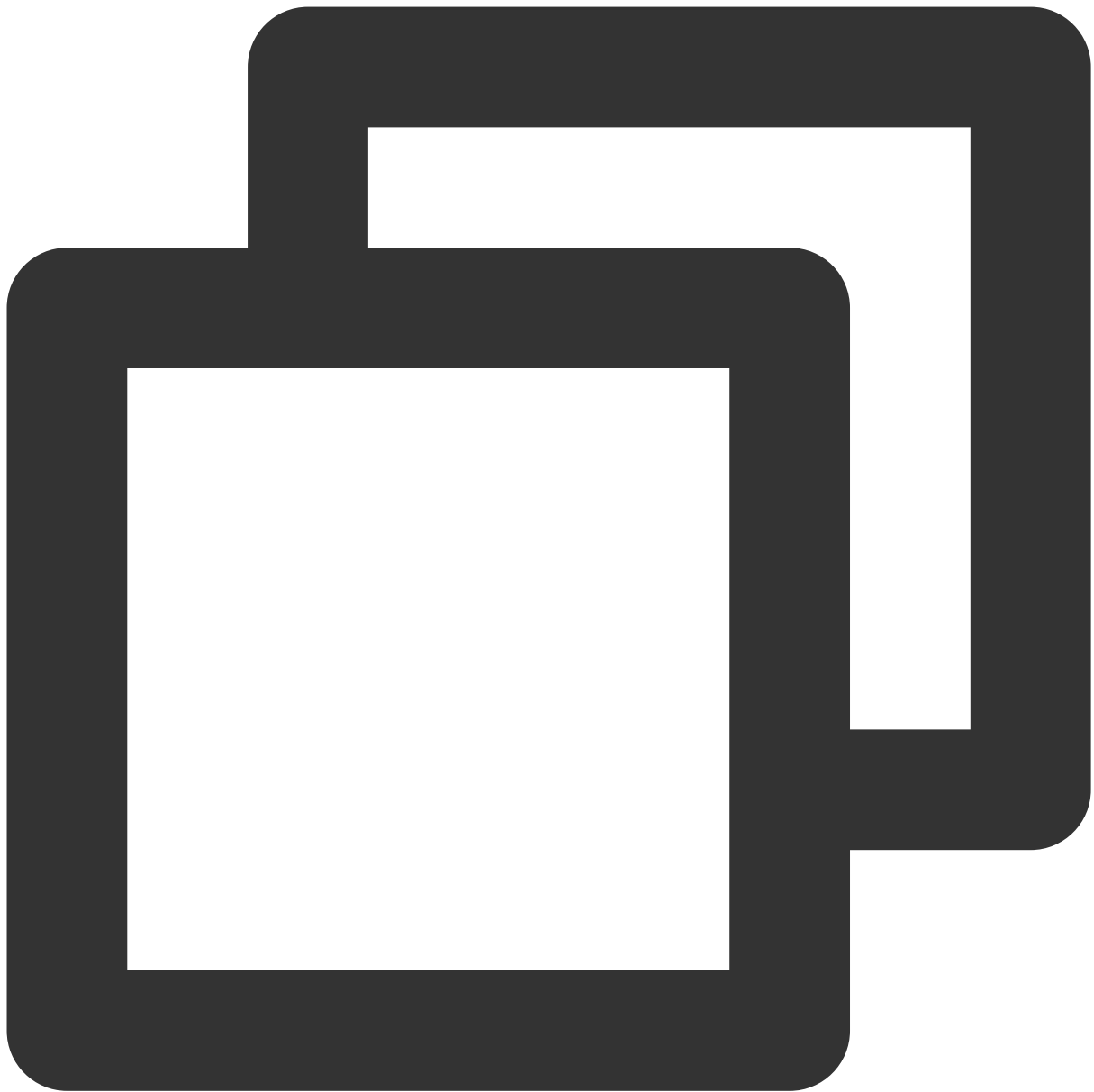
WHERE clause

LIMIT clause

COS Select currently does not support clause queries or joins.

SELECT Statement

The SELECT statement can extract the data you want to see from a COS object. You can query the data at different dimensions such as column name, function, and expression, and the query result will be returned as a list. The format of SELECT statement call is as follows:

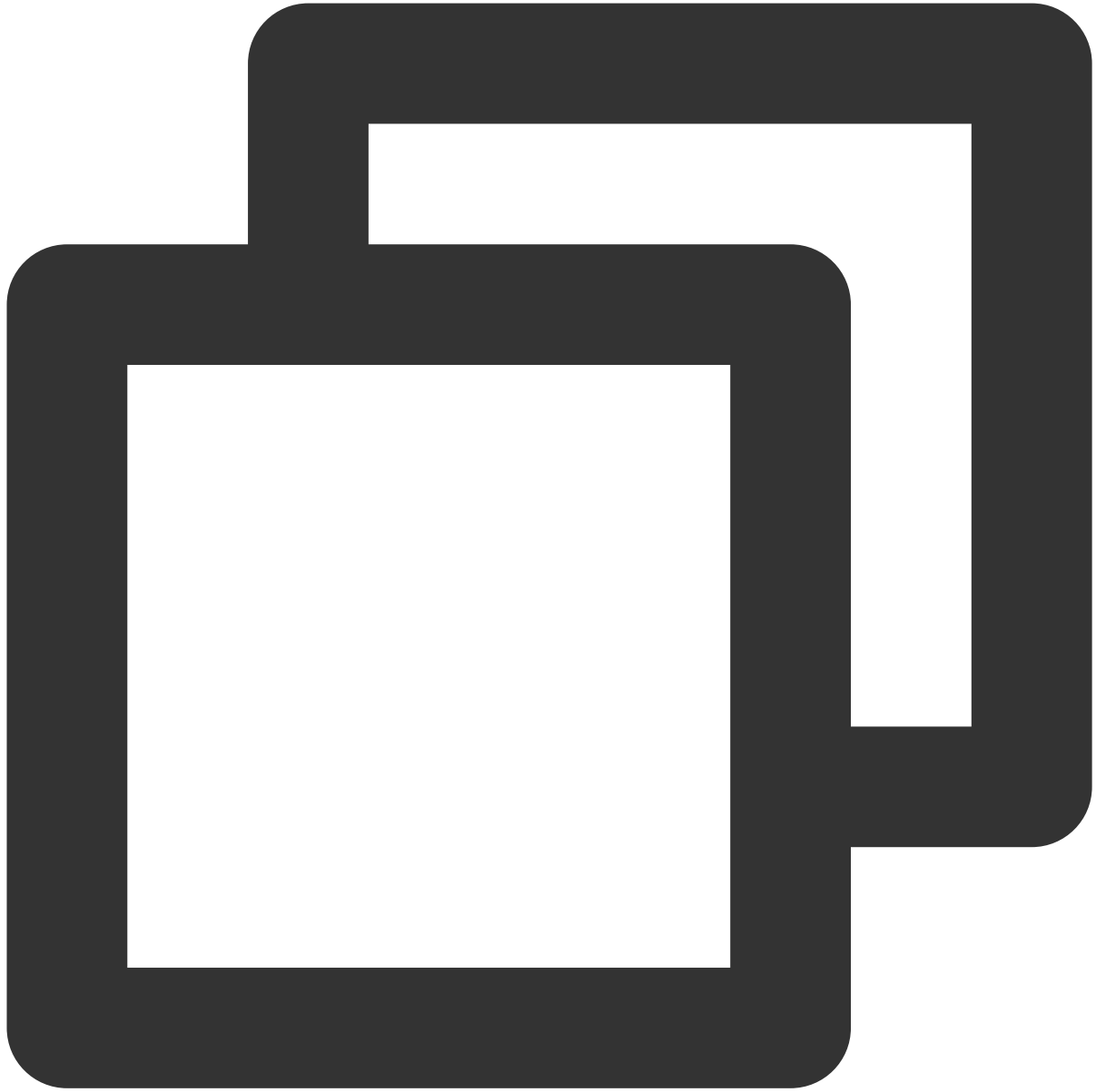


```
SELECT *  
SELECT projection [ AS column_alias | column_alias ] [, ...]
```

The first SELECT statement is marked with `*` (asterisk) and will return all the columns in the COS object. The second one uses a user-defined output scalar expression, and **projection** creates a list of outputs with custom names for each column.

WHERE Clause

The WHERE clause uses the following syntax:

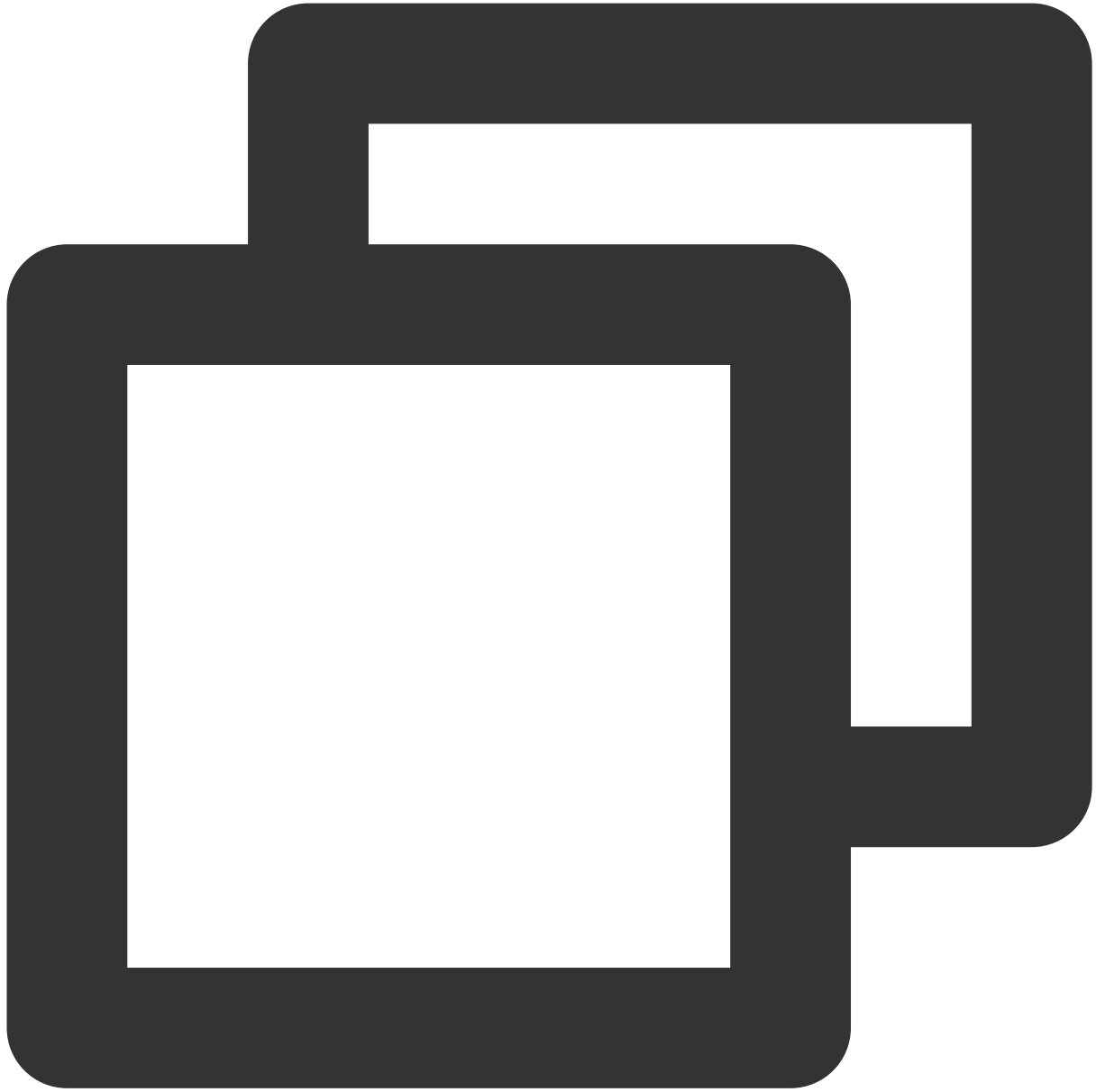


```
WHERE condition
```

The WHERE clause filters data by **condition**. **condition** is an expression that returns a Boolean result, and only rows with a return value of TRUE will be output in the result.

LIMIT Clause

The LIMIT clause uses the following syntax:



```
LIMIT number
```

The LIMIT clause sets a limit on the number of records to be returned per query, which can be specified using the **number** parameter.

Access Attributes

The SELECT and WHERE clauses can select the fields to be queried in any of the following ways, depending on whether the file format is CSV or JSON.

CSV

Column number: You can use `_N` to specify the data in column N for query. For any CSV files, the column number increases from 1. For example, the first column is numbered `_1`, and the second column is numbered `_2`. In the SELECT and WHERE clauses, it is valid to specify the column to be queried using `_N` or `alias._N`.

Column header: If the CSV file to be queried contains column headers, the SELECT and WHERE clauses can use the headers to specify the columns to be queried, which can be specified using `alias.column_name` or `column_name` in the SELECT and WHERE clauses in an SQL statement.

JSON

Document: You can access a JSON document using `alias.name`. A nested array can be accessed in a way such as `alias.name1.name2.name3`.

List: You can access the elements in a list using an index, which is numbered from 0 and uses the `[]` operator. For example, you can access the second element in a JSON list using `alias[1]`. If you need to access a nested array, you can also do so in a way such as `alias.name1.name2[1].name3`.

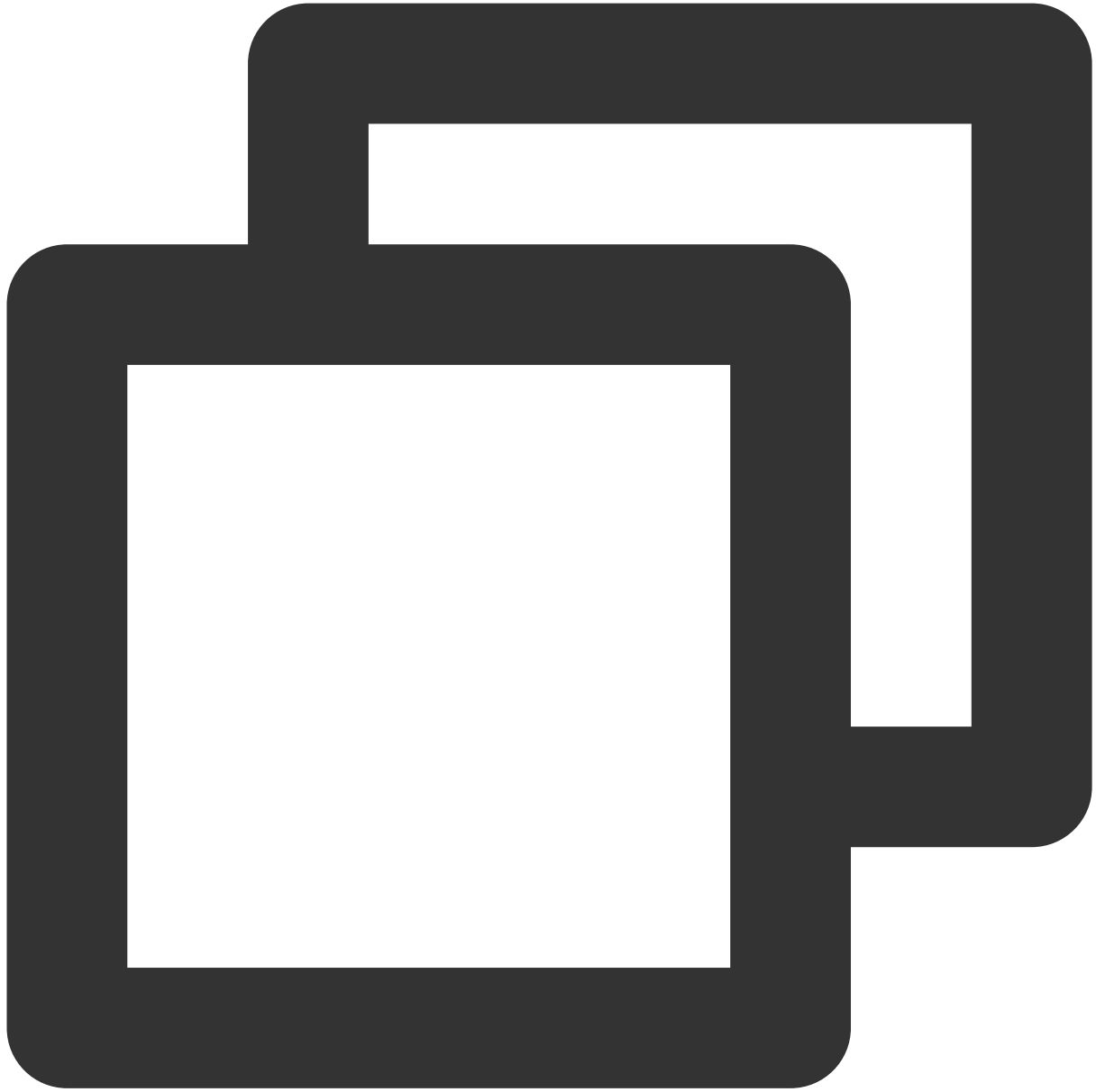
Samples

The following is the sample data in the samples:



```
{
  "name": "Leon",
  "org": "Tencent",
  "projects":
  [
    {"project_name":"project1", "completed":true},
    {"project_name":"project2", "completed":false}
  ]
}
```


Sample 1. The following is the SQL statement used to query `name` in the sample data and the query result:



```
Select s.name from COSObject s
```



```
{"name": "Leon"}
```

Sample 2. The following is the SQL statement used to query `project_name` in the sample data and the query result:



```
Select s.projects[0].project_name from COSObject s
```



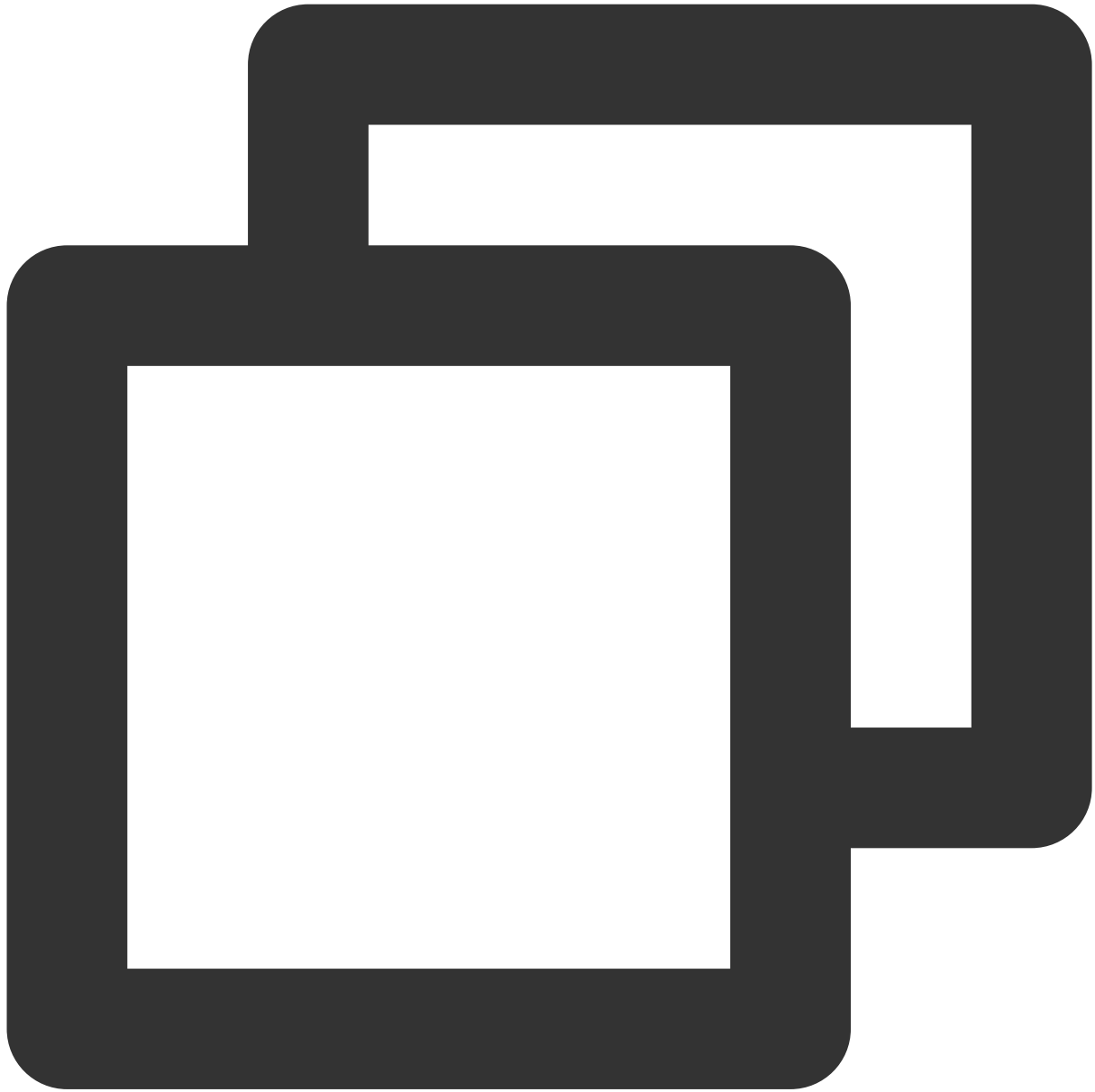
```
{"project_name": "project1"}
```

Case Sensitivity of Headers and Attribute Names

You can use double quotation marks to indicate whether headers in a CSV file and attribute names in a JSON file are case-sensitive. If no double quotation marks are added, the headers/attribute names are case-insensitive. If this is not explicitly specified, COS Select may throw an exception.

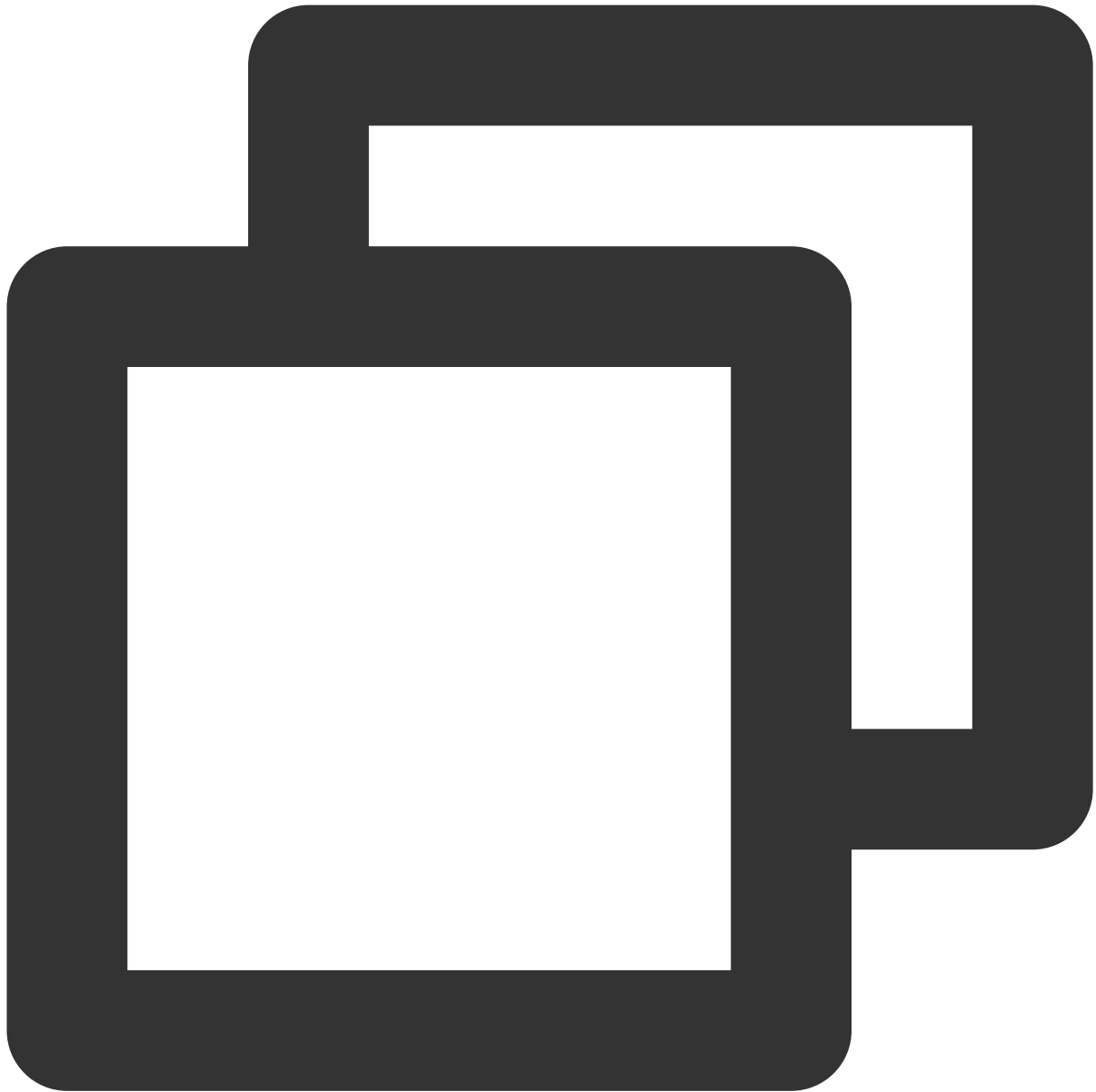
Sample 1. Query objects with a header/attribute name containing "NAME".

Because the following sample SQL query does not contain double quotation marks, the query is case-insensitive. As this header is present in the table, a value will be successfully returned eventually.



```
SELECT s.name from COSObject s
```

Because the following sample SQL query contains double quotation marks, the query is case-sensitive. As the table does not contain this header, the `SQLParsingError` 400 error will be eventually returned.



```
SELECT s."name" from COSObject s
```

Sample 2. Query objects with a header/attribute name containing "NAME" and "name".

Because the following sample SQL query does not contain double quotation marks, the query is case-insensitive. As the table contains two headers "NAME" and "name", the query command is ambiguous, and the

`AmbiguousFieldName` exception will be thrown.



```
SELECT s.name from COSObject s
```

Because the following sample SQL query contains double quotation marks, the query is case-sensitive. As the table contains the header "NAME", the query result will be successfully returned.



```
SELECT s."NAME" from COSObject s
```

Using Reserved Fields as User-defined Fields

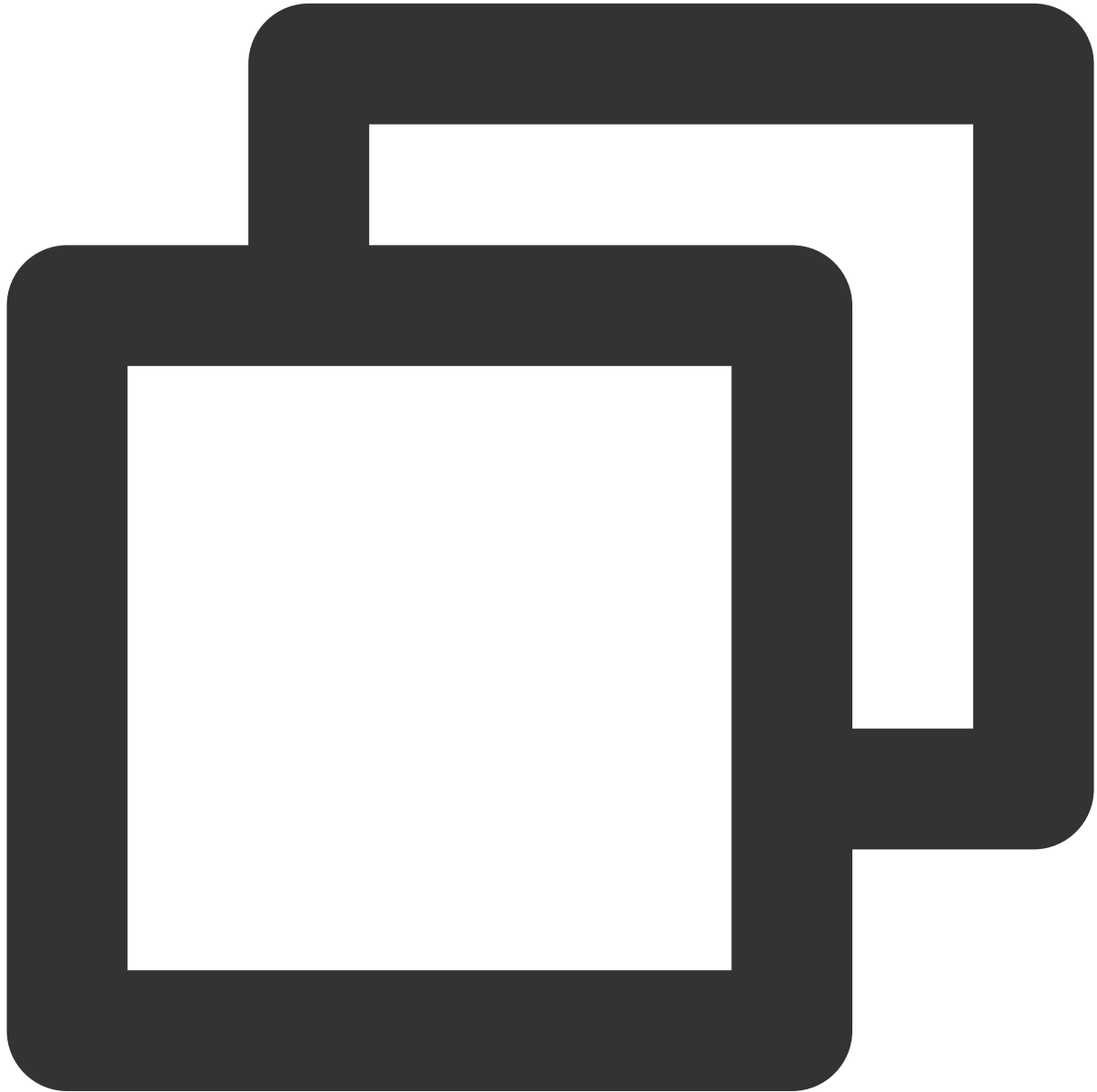
The SQL expressions of COS Select have certain reserved fields such as function name, data type, and operator. Sometimes you probably use these reserved fields as column headers in a CSV file or attribute names in a JSON,

which may cause conflicts with reserved fields. In this case, you can use double quotation marks to indicate that you are using a custom field; otherwise, COS will return `400 parse error`.

For the complete list of reserved fields, see [Reserved Words](#).

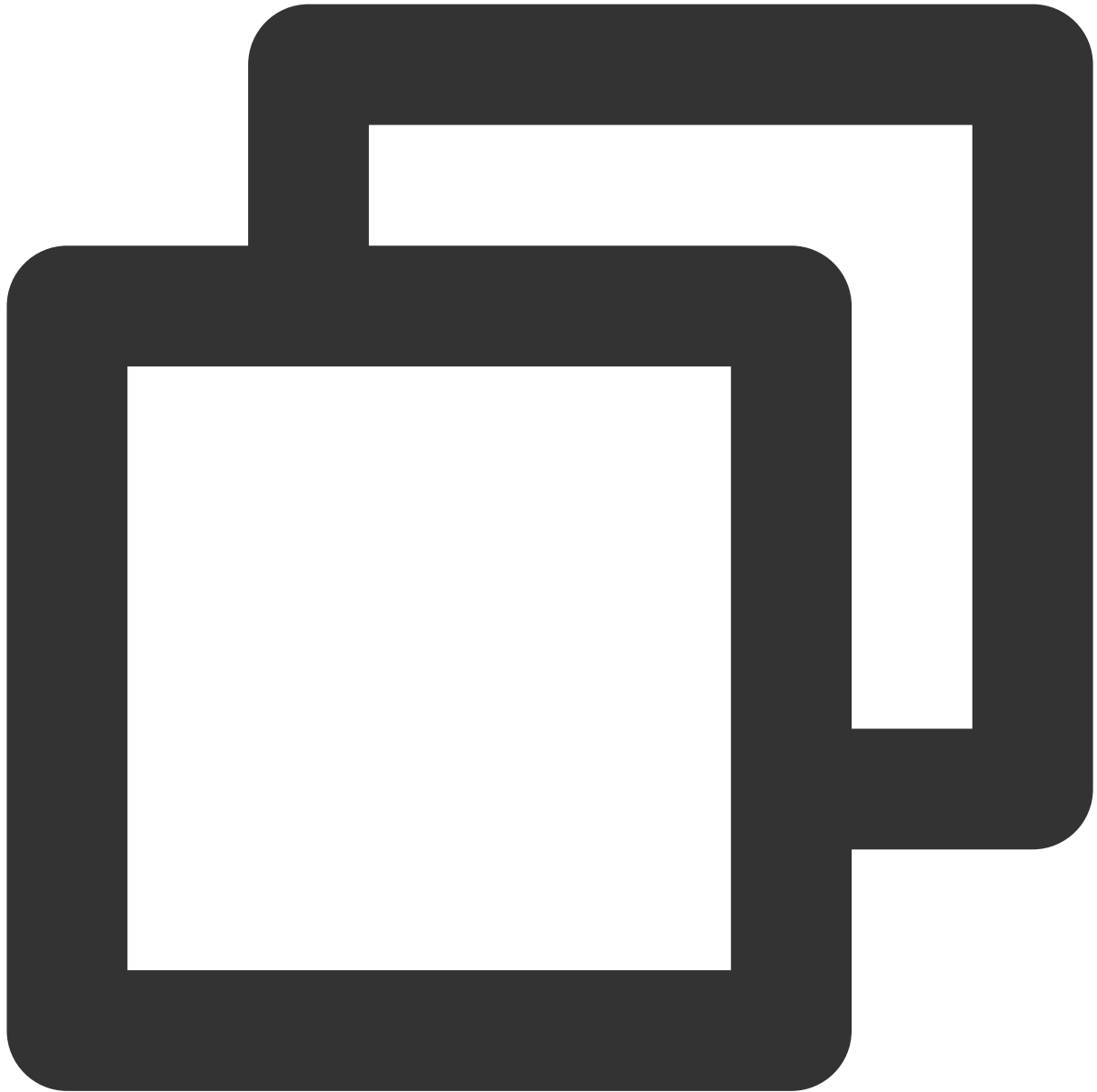
Sample: The header/attribute name in the object to be queried contains a reserved field "CAST".

The following sample SQL query uses double quotation marks to indicate that CAST is a user-defined field, so the query result will be successfully returned.



```
SELECT s."CAST" from COSObject s
```

The following sample SQL query does not use double quotation marks to indicate that CAST is a user-defined field, so COS will treat it as a reserved field and return `400 parse error`.



```
SELECT s.CAST from COSObject s
```

Scalar Expressions

In the SELECT statement and the WHERE clause, you can use SQL scalar expressions (expressions that return a scalar). Currently, COS Select supports the following forms:

literal: SQL text.

column_reference: column_name or alias.column_name.

unary_opexpression: SQL unary operator.

expressionbinary_opexpression: SQL binary operator.

func_name: Name of the called scalar function.

expression [NOT] BETWEEN expression AND expression

expression LIKE expression [ESCAPE expression]

SQL Functions

Last updated : 2024-03-25 15:28:24

Aggregate Functions

COS Select supports the following aggregate functions:

Function Name	Parameter Type	Return Type
AVG(expression)	INT, FLOAT, and DECIMAL	DECIMAL will be returned if the input parameter is of integer type, and FLOAT if float type. The same type as the input parameter will be returned in all other cases.
COUNT	-	INT
MAX(expression)	INT and DECIMAL	The return value type is the same as that of the input parameter
MIN(expression)	INT and DECIMAL	The return value type is the same as that of the input parameter
SUM(expression)	INT, FLOAT, DOUBLE, and DECIMAL	INT will be returned if the input parameter is of integer type, and FLOAT if float type. The same type as the input parameter will be returned in all other cases.

Condition Functions

COS Select supports the following condition functions:

COALESCE

The COALESCE function determines the input parameters in sequence and returns the first non-null parameter value. If the input parameters do not include a non-null parameter, the function will return a null value.

Syntax



```
COALESCE ( expression, expression, ... )
```

Note:

Values, arrays, or nested functions of INT, String, and Float types can be passed in for the `expression` parameter.

Sample

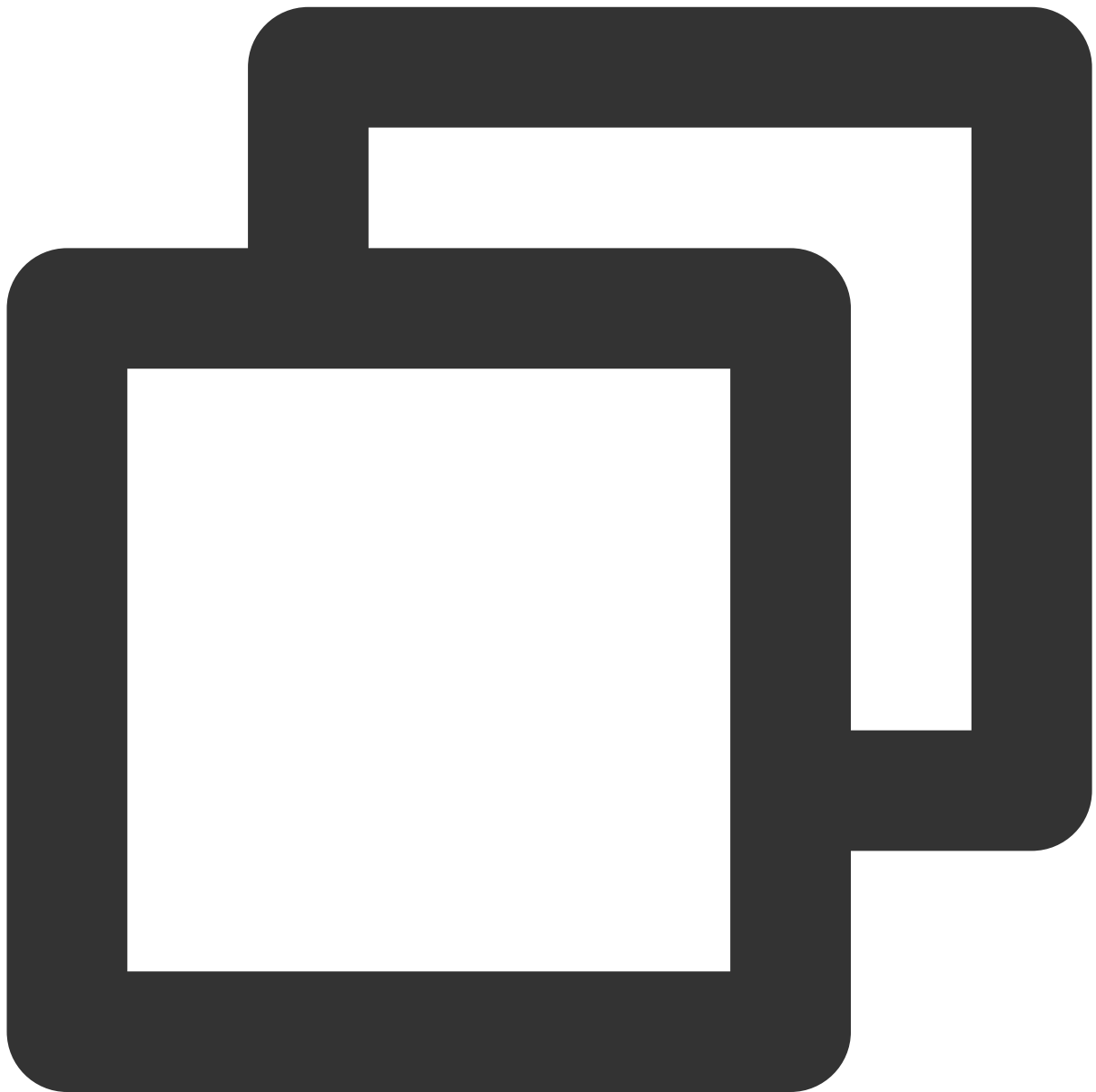


```
COALESCE(1) -- 1
COALESCE(1, null) -- 1
COALESCE(null, null, 1) -- 1
COALESCE(missing, 1) -- 1
COALESCE(null, 'string') -- 'string'
COALESCE(null) -- null
COALESCE(null, null) -- null
COALESCE(missing) -- null
COALESCE(missing, missing) -- null
```

NULLIF

The NULLIF function determines the difference between two parameters passed in. If the two parameters have the same value, NULL will be returned; otherwise, the value of the first parameter passed in will be returned.

Syntax

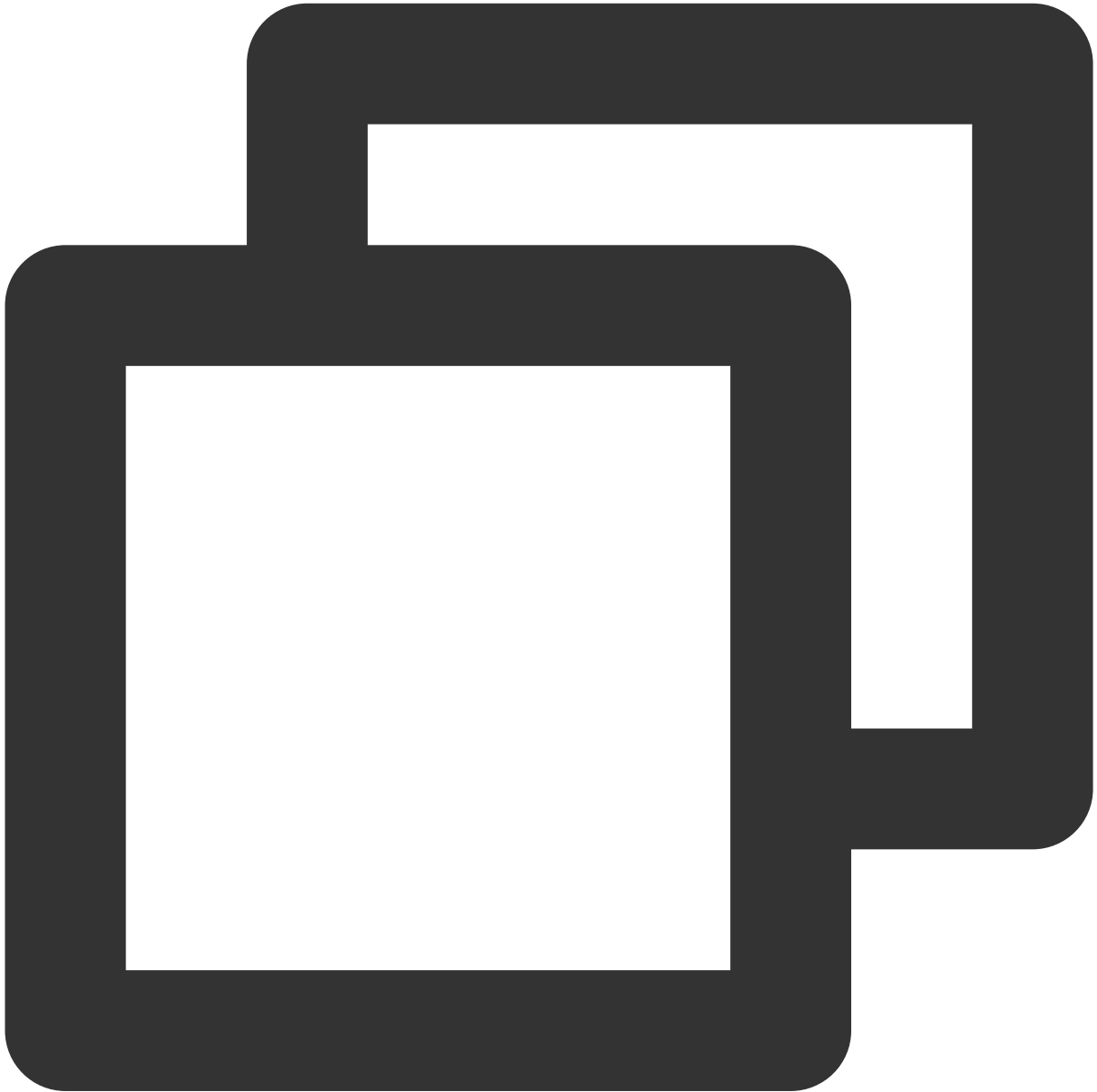


```
NULLIF ( expression1, expression2 )
```

Note:

Values, arrays, or nested functions of INT, String, and Float types can be passed in for the `expression` parameter.

Sample



```
NULLIF(1, 2)           -- 1
NULLIF(1, '1')         -- 1
NULLIF(1, NULL)        -- 1
NULLIF(1, 1)           -- null
NULLIF(1.0, 1)         -- null
NULLIF(missing, null)  -- null
```



```
NULLIF(missing, missing) -- null
NULLIF([1], [1])         -- null
NULLIF(NULL, 1)           -- null
NULLIF(null, null)       -- null
```

Conversion Functions

COS Select supports the following conversion functions:

CAST

The CAST function converts one instance to another instance. The instance can be either a value or a function that can be calculated to a certain value.

Syntax



```
CAST ( expression AS data_type )
```

Note:

The `expression` parameter can be a value, an array, an operator, or an SQL function that can be calculated to a certain value.

The `data_type` parameter is the data type after conversion, such as INT. For the data types currently supported by COS Select, see [Data Types](#).

Sample



```
CAST('2007-04-05T14:30Z' AS TIMESTAMP)  
CAST(0.456 AS FLOAT)
```

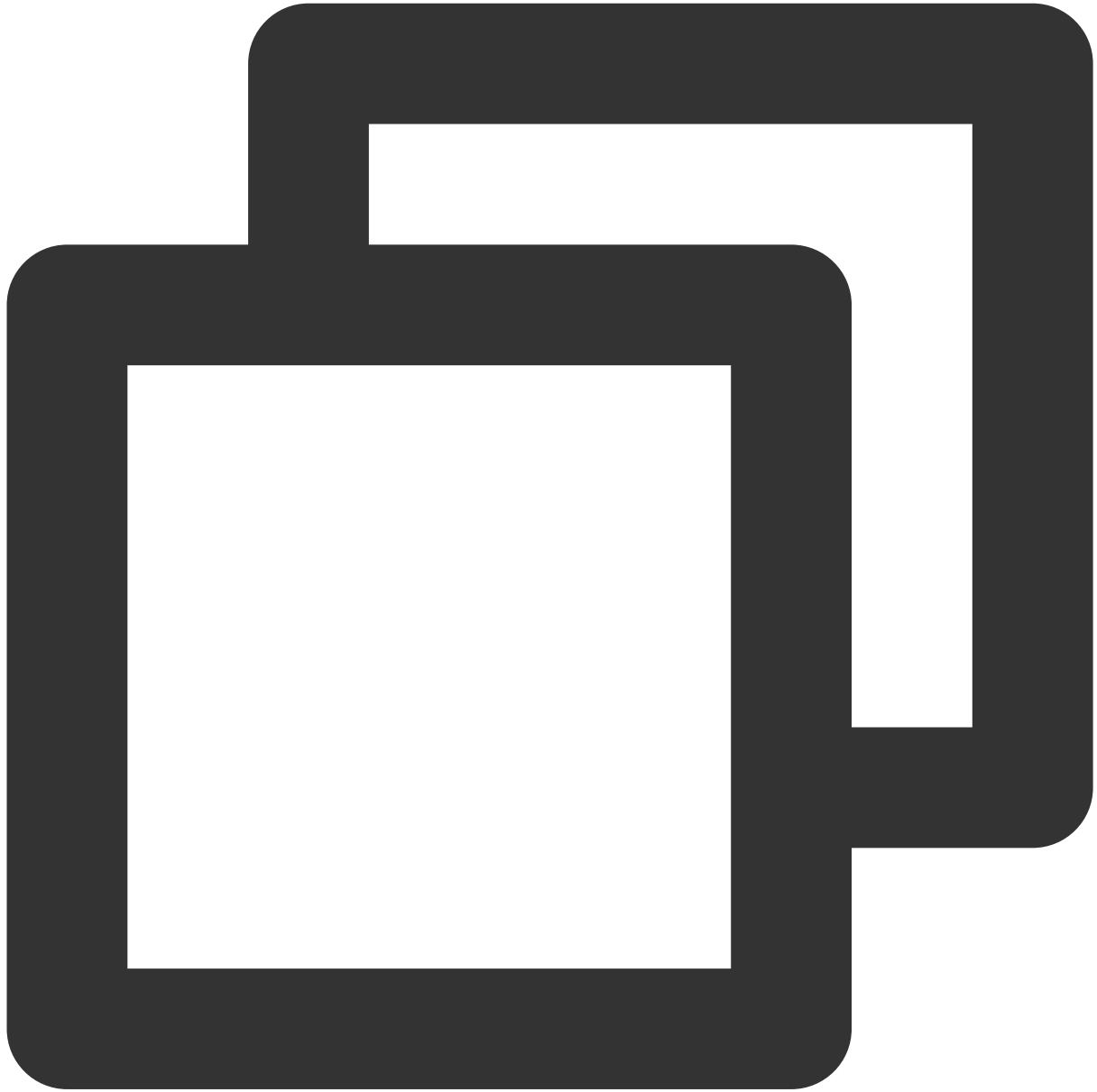
Date Functions

COS Select supports the following date functions:

DATE_ADD

The `DATE_ADD` function adds a specified time interval to a part (year, month, day, hour, minute, or second) of a specific timestamp and returns a new timestamp.

Syntax



```
DATE_ADD( date_part, quantity, timestamp )
```

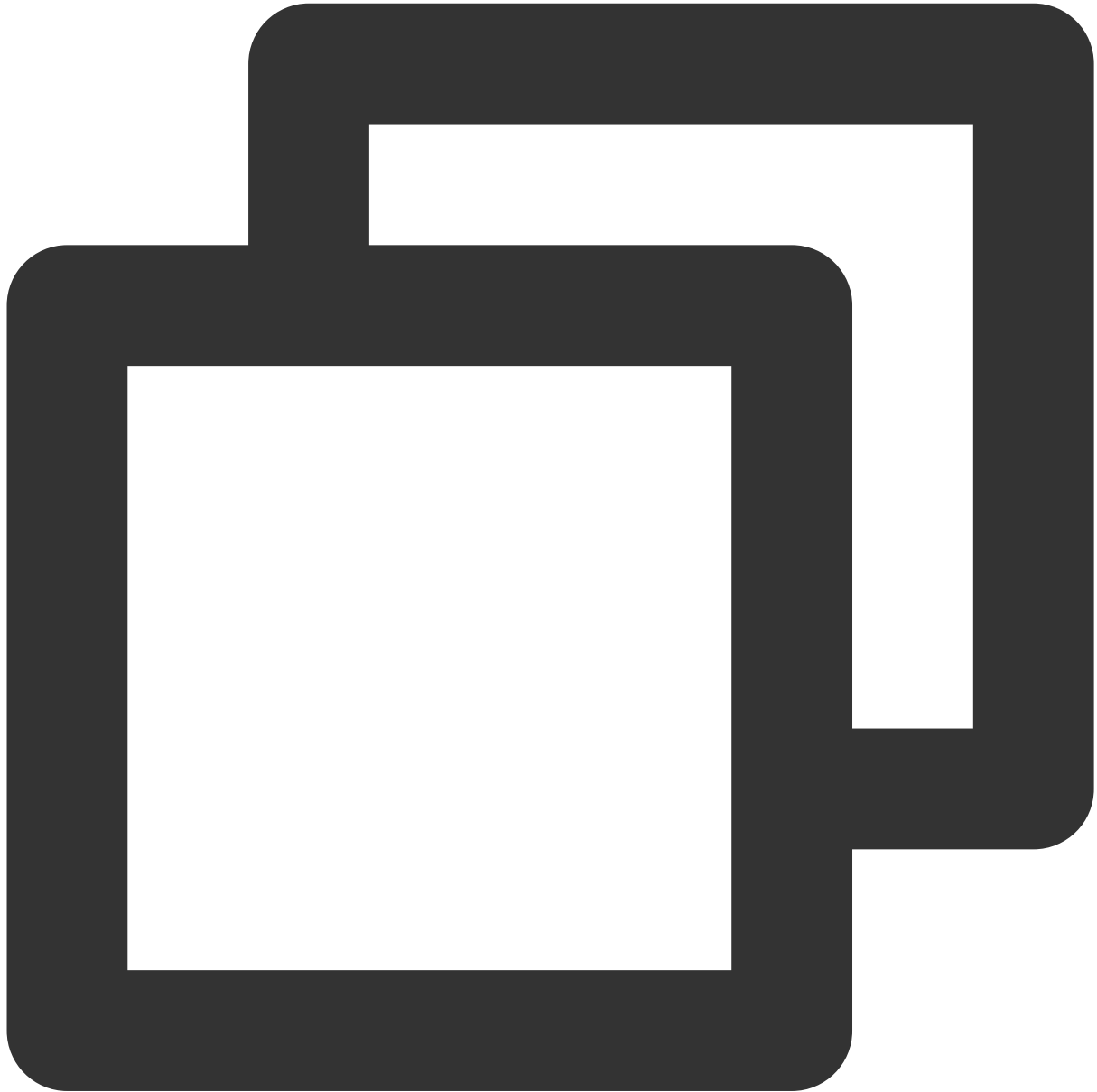
Note:

The `date_part` parameter specifies the part of the timestamp to be modified, which can be year, month, day, hour, minute, or second.

The `quantity` parameter represents the value to be added, which must be a positive integer.

The `timestamp` parameter represents the timestamp to be modified.

Sample



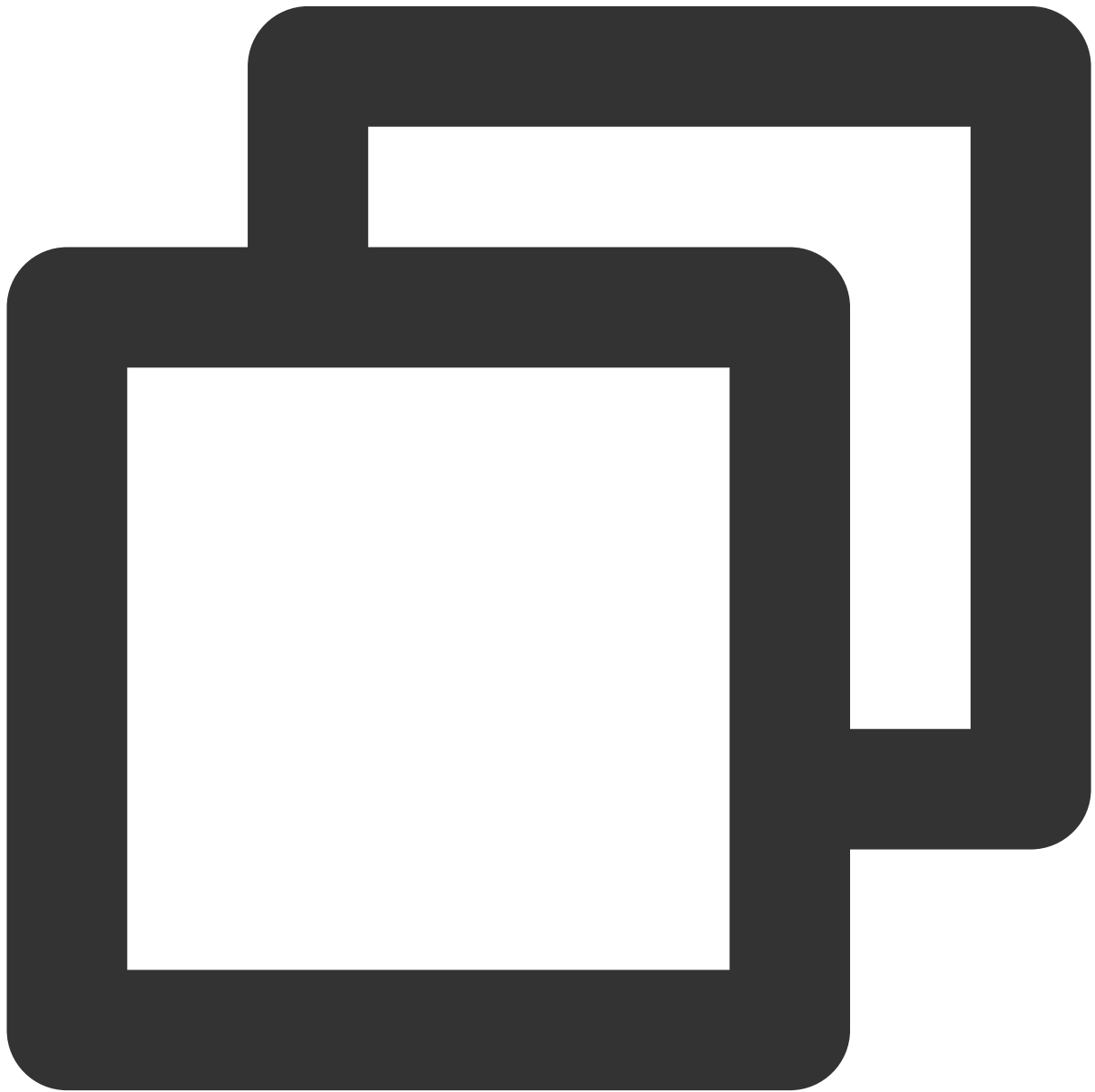
```
DATE_ADD(year, 5, `2010-01-01T`) -- 2015-01-01
DATE_ADD(month, 1, `2010T`) -- 2010-02T
DATE_ADD(month, 13, `2010T`) -- 2011-02T
DATE_ADD(hour, 1, `2017T`) -- 2017-01-01T01:00-00:00
DATE_ADD(hour, 1, `2017-01-02T03:04Z`) -- 2017-01-02T04:04Z
DATE_ADD(minute, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:05:05.006Z
```

```
DATE_ADD(second, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:04:06.006Z
```

DATE_DIFF

The DATE_DIFF function compares two valid timestamps and returns the difference between them, which can be displayed in the specified unit of time. If the `date_part` value of timestamp1 is greater than that of timestamp2, a positive number will be returned; otherwise, a negative number will be returned.

Syntax



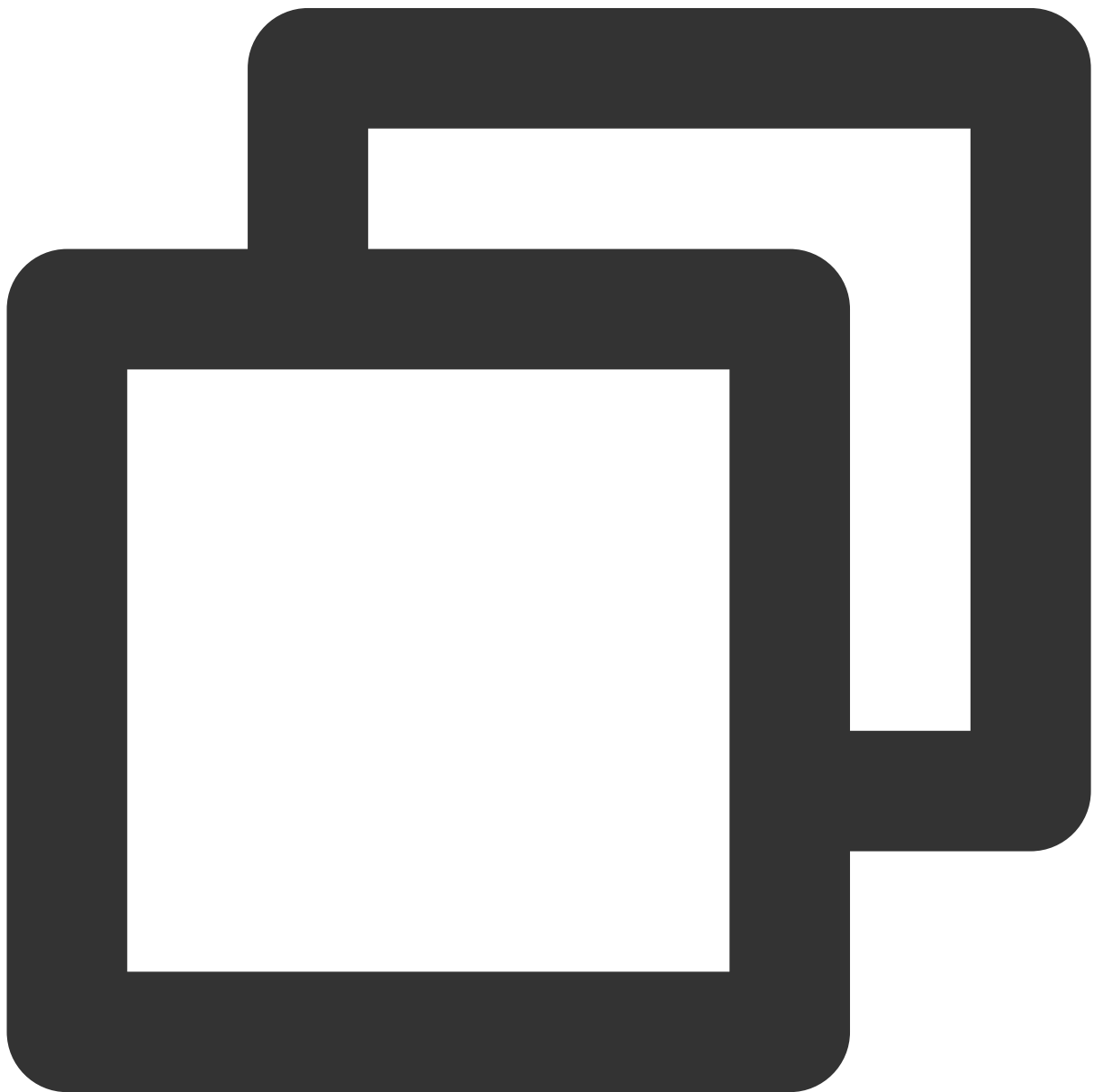
```
DATE_DIFF( date_part, timestamp1, timestamp2 )
```

Note:

The `date_part` parameter specifies the unit of time which the two timestamps are compared in and can be year, month, day, hour, minute, or second.

The `timestamp1` parameter is the first input timestamp.

The `timestamp2` parameter is the second input timestamp.

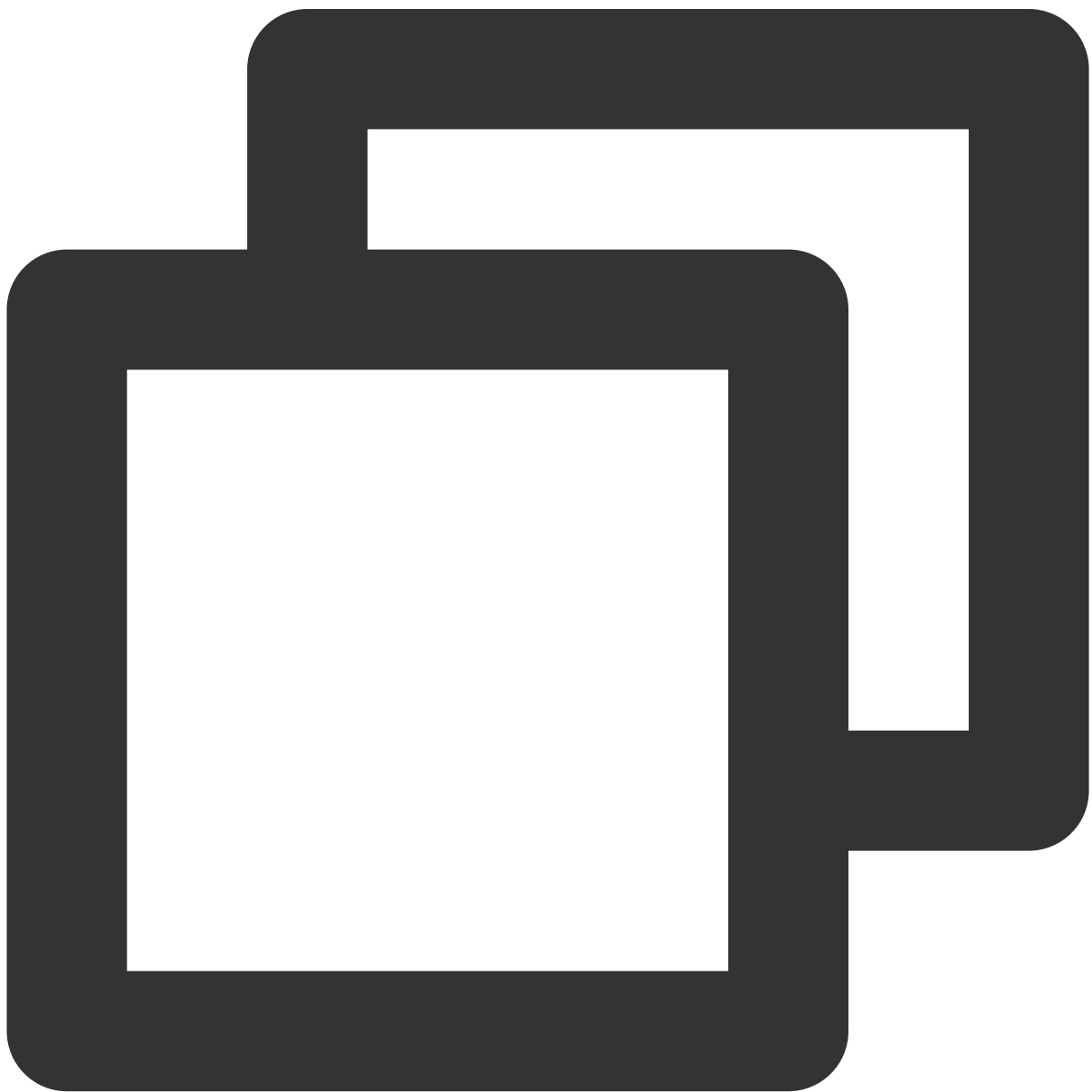
Sample

```
DATE_DIFF(year, `2010-01-01T`, `2011-01-01T`)      -- 1
DATE_DIFF(year, `2010T`, `2010-05T`)                -- 4
DATE_DIFF(month, `2010T`, `2011T`)                  -- 12
DATE_DIFF(month, `2011T`, `2010T`)                  -- -12
DATE_DIFF(day, `2010-01-01T23:00T`, `2010-01-02T01:00T`) -- 0
```

EXTRACT

The EXTRACT function extracts a value in the specified unit of time from a given timestamp.

Syntax

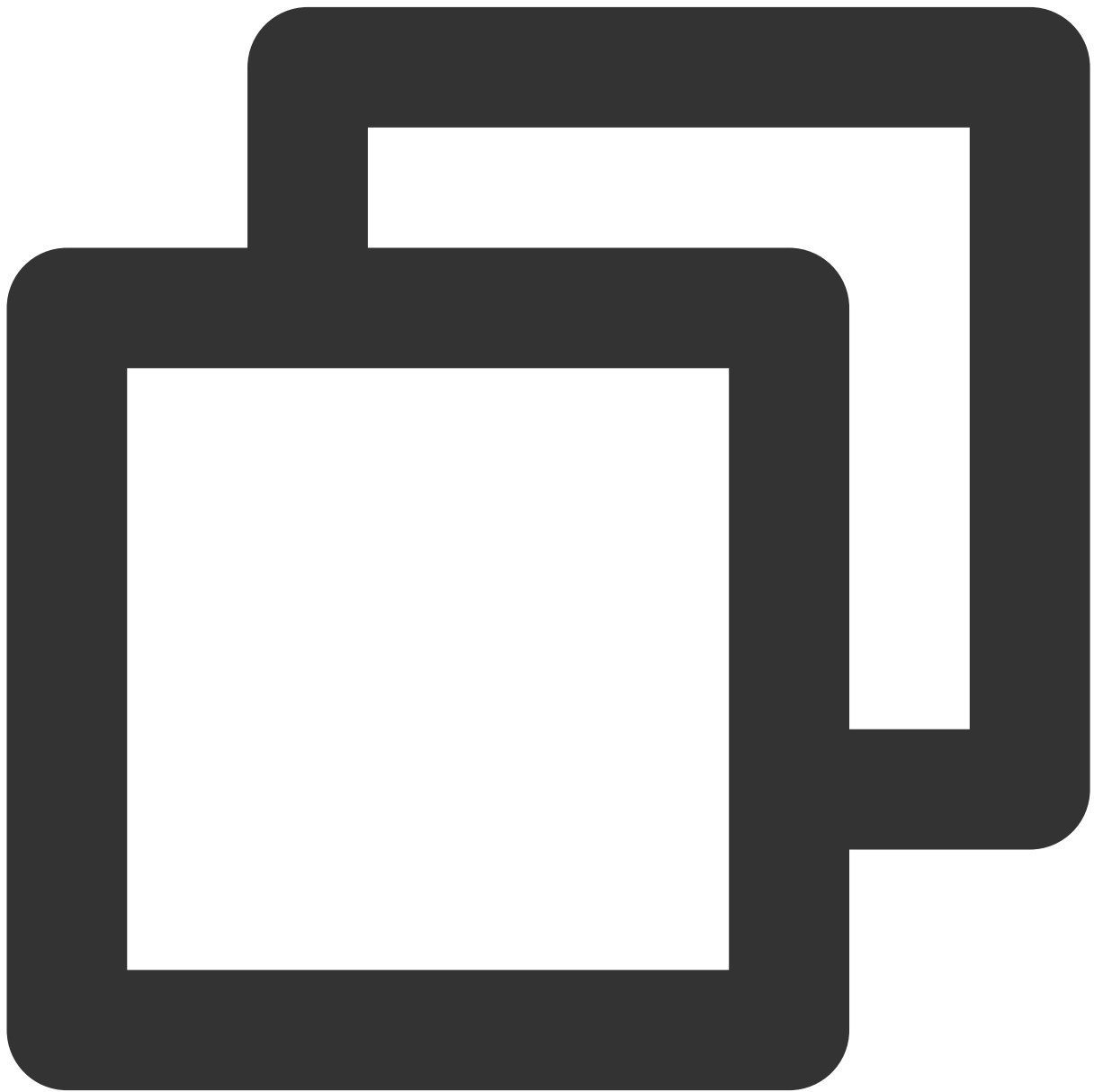



```
EXTRACT( date_part FROM timestamp )
```

Note:

The parameter `date_part` specifies the unit of time to be extracted, which can be year, month, day, hour, minute, or second.

The `timestamp` parameter represents the input timestamp.

Sample

```
EXTRACT(YEAR FROM `2010-01-01T`)
```

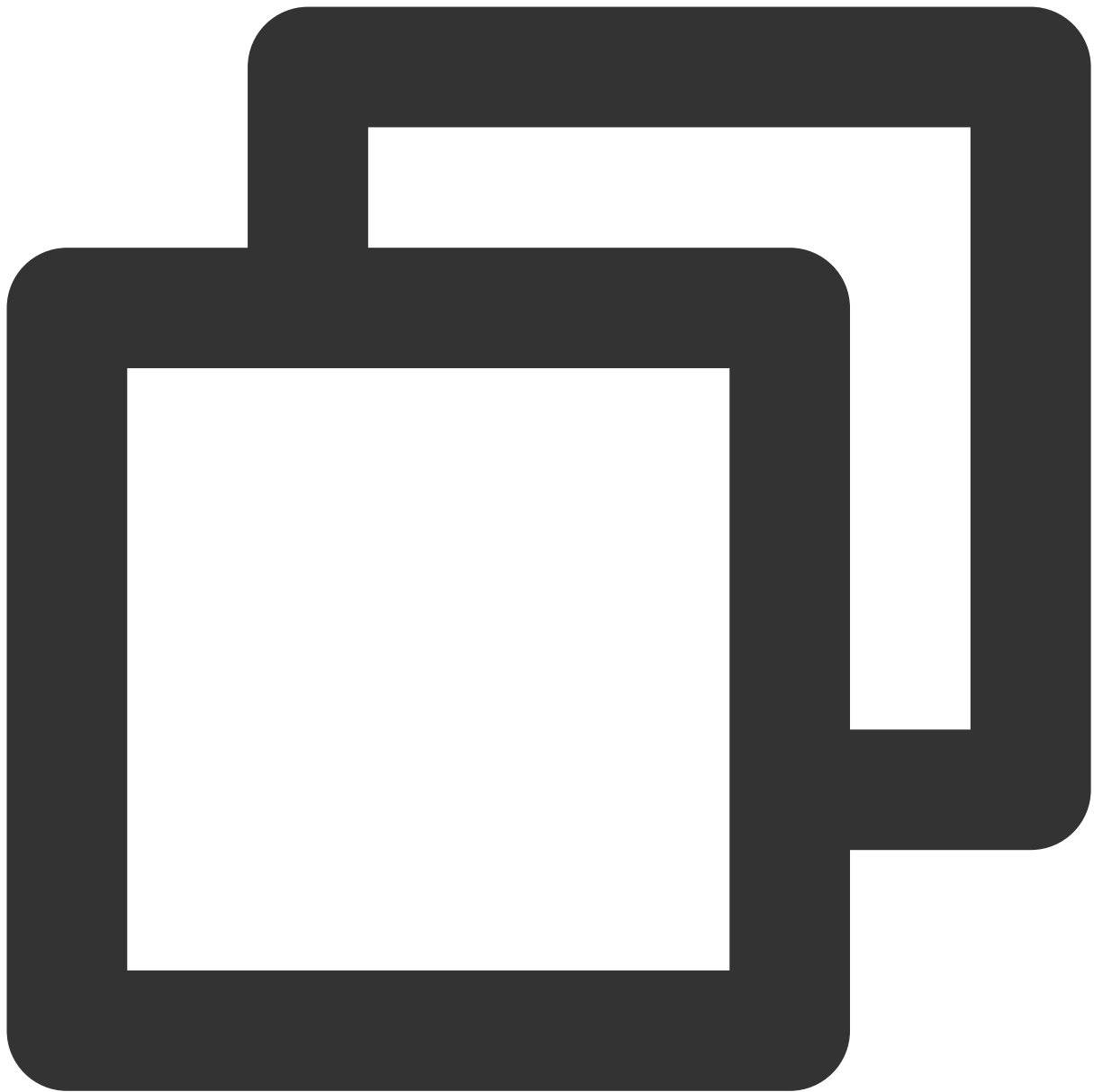
```
-- 2010
```

```
EXTRACT(MONTH FROM `2010T`) -- 1
EXTRACT(MONTH FROM `2010-10T`) -- 10
EXTRACT(HOUR FROM `2017-01-02T03:04:05+07:08`) -- 3
EXTRACT(MINUTE FROM `2017-01-02T03:04:05+07:08`) -- 4
EXTRACT(TIMEZONE_HOUR FROM `2017-01-02T03:04:05+07:08`) -- 7
EXTRACT(TIMEZONE_MINUTE FROM `2017-01-02T03:04:05+07:08`) -- 8
```

TO_STRING

The TO_STRING function converts a timestamp to a string of time in the specified format.

Syntax



```
TO_STRING ( timestamp time_format_pattern )
```

Note:

The `timestamp` parameter specifies the timestamp to be converted.

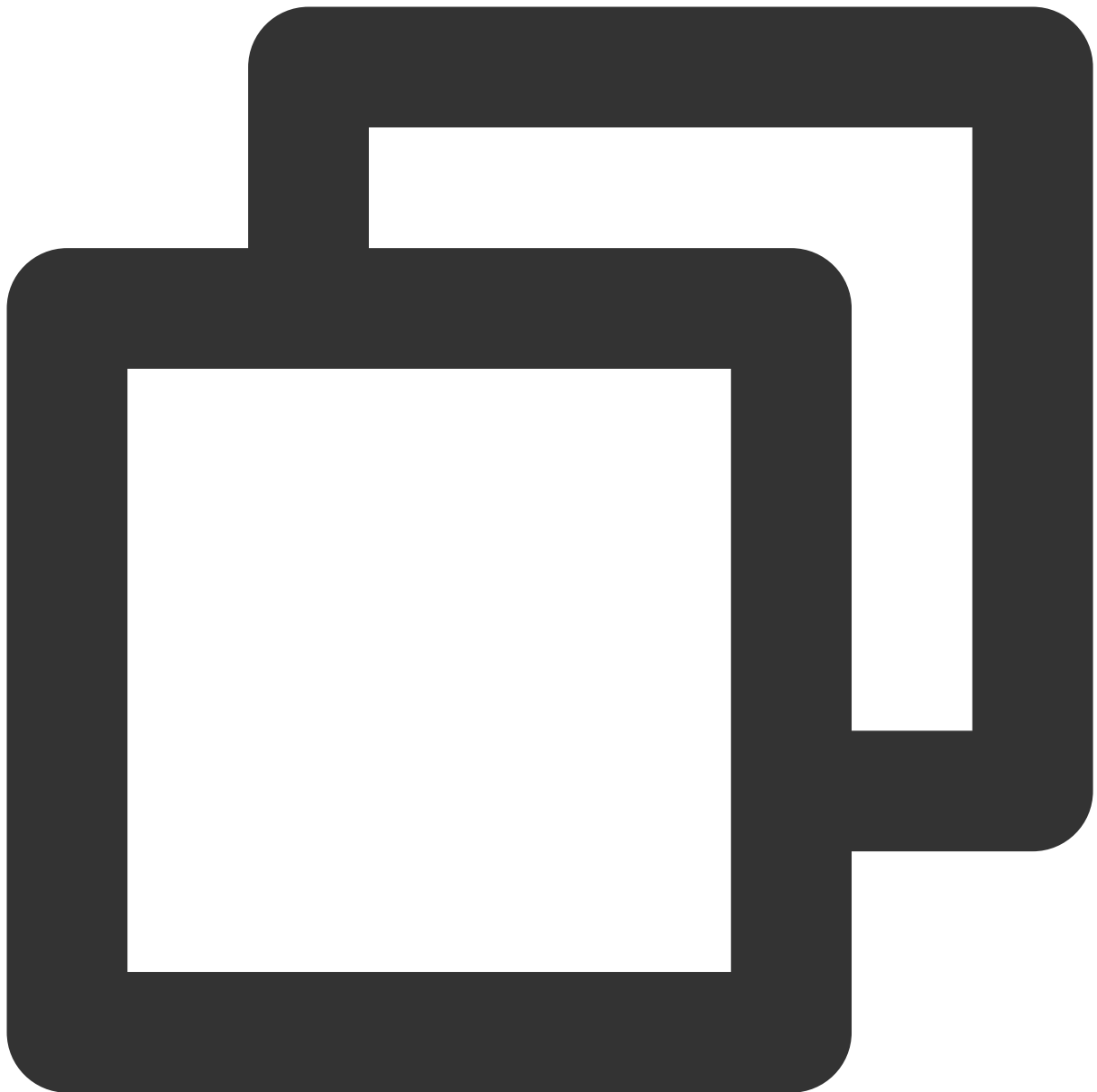
The `time_format_pattern` parameter specifies the time format.

Format	Description	Sample
yy	2-digit year	98
y	4-digit year	1998

yyyy	Year expressed by 4 digits. If there are less than 4 digits, 0 will be automatically added	0199
M	Month	1
MM	Month expressed by 2 digits. If there are less than 2 digits, 0 will be automatically added	01
MMM	English abbreviation of a month	Jan
MMMM	Full English name of a month	January
MMMMM	Initial of a month	J (not applicable to the to_timestamp function)
d	Day (1-31) in a month	1
dd	Day expressed by 2 digits (1-31)	01
a	Symbol for morning or afternoon (AM/PM)	AM
h	Hour in 12-hour time	1
hh	Hour expressed by 2 digits in 12-hour time	01
H	Hour in 24-hour time	1
HH	Hour expressed by 2 digits in 24-hour time	01
m	Minute (00-59)	1
mm	Minute expressed by 2 digits in 24-hour time	01
s	Second (00-59)	1
ss	Second expressed by 2 digits in 24-hour time	01
S	Decimal part of the second (accuracy: 0.1; value range: 0.0 - 0.9)	0
SS	Decimal part of the second (accuracy: 0.01; value range: 0.00 - 0.99)	6
SSS	Decimal part of the second (accuracy: 0.001; value range: 0.000 - 0.999)	60
...
SSSSSSSSS	Decimal part of the second (accuracy: 0.000000001; value	60000000

	range: 0.0000000000 - 0.9999999999)	
n	Nanosecond	600000000
X	Hour-level offset. If the offset is 0, then this will be "Z"	+01 or Z
XX or XXXX	Hour- or minute-level offset. If the offset is 0, then this will be "Z"	+0100 or Z
xxx or xxxxx	Hour- or minute-level offset. If the offset is 0, then this will be "Z"	+01:00 or Z
x	Hour-level offset	1
xx or xxxx	Hour- or minute-level offset	0100
xxx or xxxxx	Hour- or minute-level offset	01:00

Sample

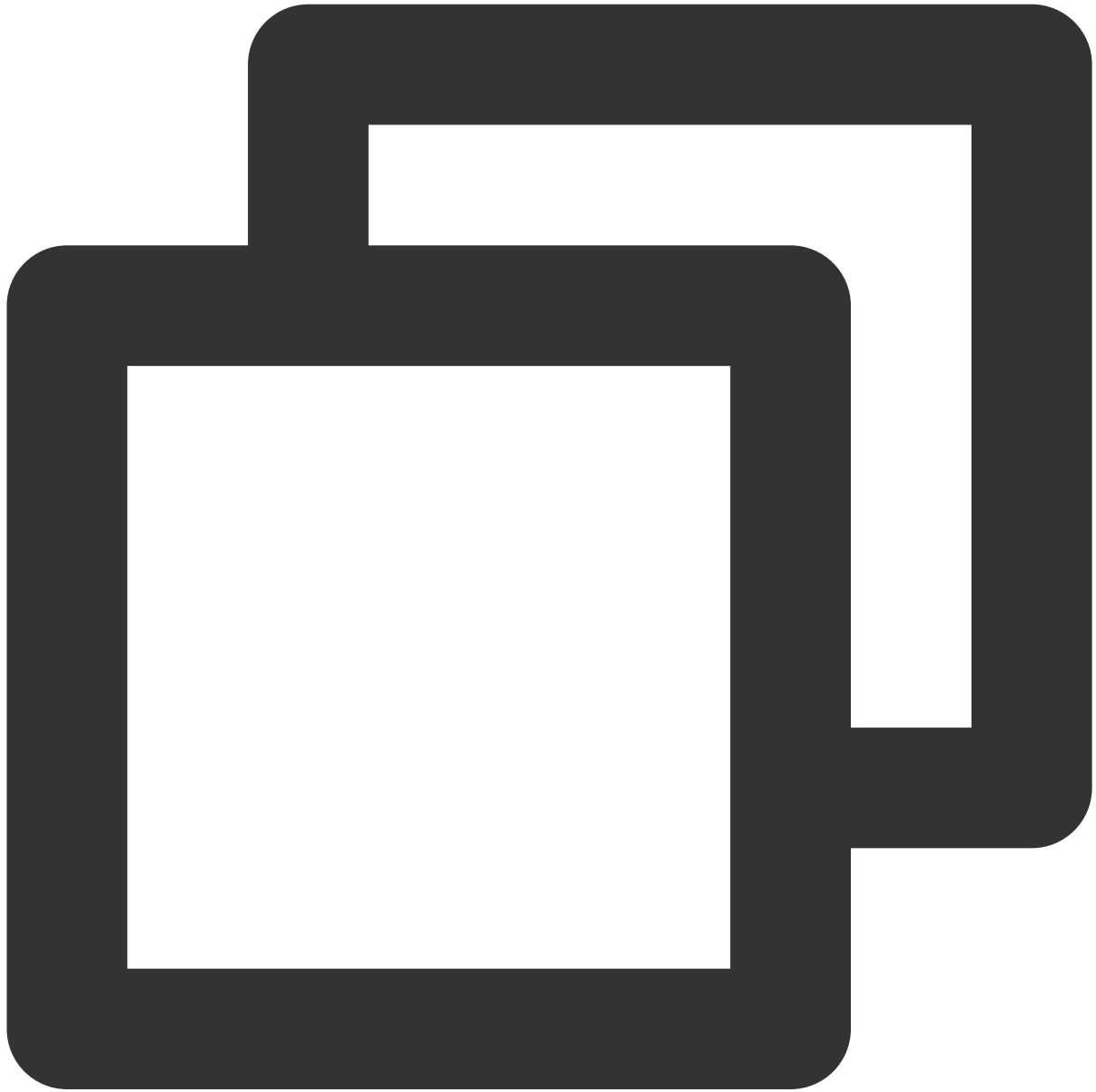


```
TO_STRING(`1998-07-20T20:18Z`, 'MMMM d, y')           -- "July 20, 1998"
TO_STRING(`1998-07-20T20:18Z`, 'MMM d, yyyy')         -- "Jul 20, 1998"
TO_STRING(`1998-07-20T20:18Z`, 'M-d-yy')              -- "7-20-99"
TO_STRING(`1998-07-20T20:18Z`, 'MM-d-y')              -- "07-20-1998"
TO_STRING(`1998-07-20T20:18Z`, 'MMMM d, y h:m a')     -- "July 20, 1998 8
TO_STRING(`1998-07-20T20:18Z`, 'y-MM-dd'T'H:m:ssX')  -- "1998-07-20T20:1
TO_STRING(`1998-07-20T20:18+08:00Z`, 'y-MM-dd'T'H:m:ssX') -- "1998-07-20T20:1
TO_STRING(`1998-07-20T20:18+08:00`, 'y-MM-dd'T'H:m:ssXXXX') -- "1998-07-20T20:1
TO_STRING(`1998-07-20T20:18+08:00`, 'y-MM-dd'T'H:m:ssXXXXX') -- "1998-07-20T20:1
```

TO_TIMESTAMP

The TO_TIMESTAMP function converts a string of time to a timestamp.

Syntax

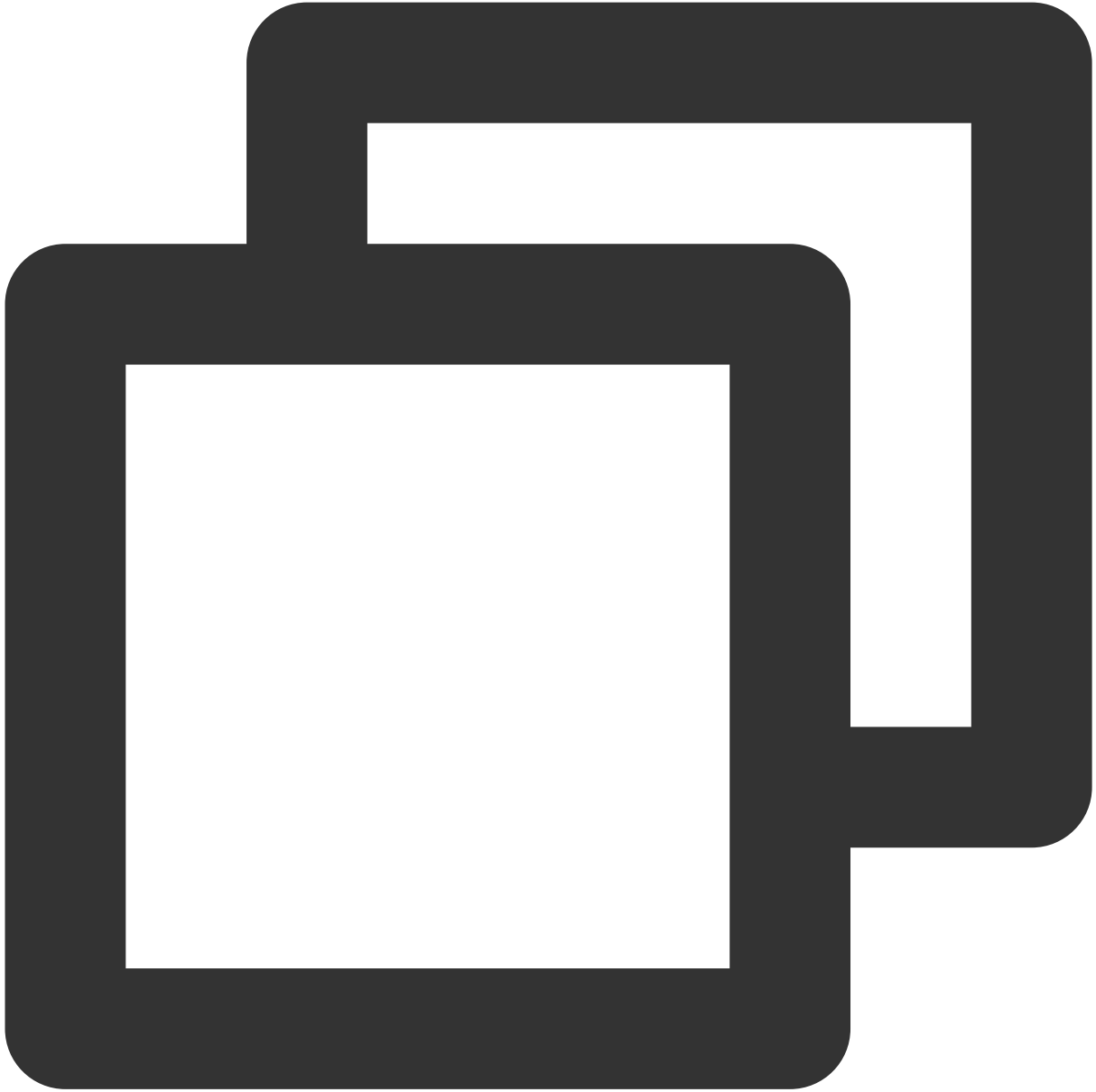


```
TO_TIMESTAMP ( string )
```

Note:

The `string` parameter represents the input time string.

Sample

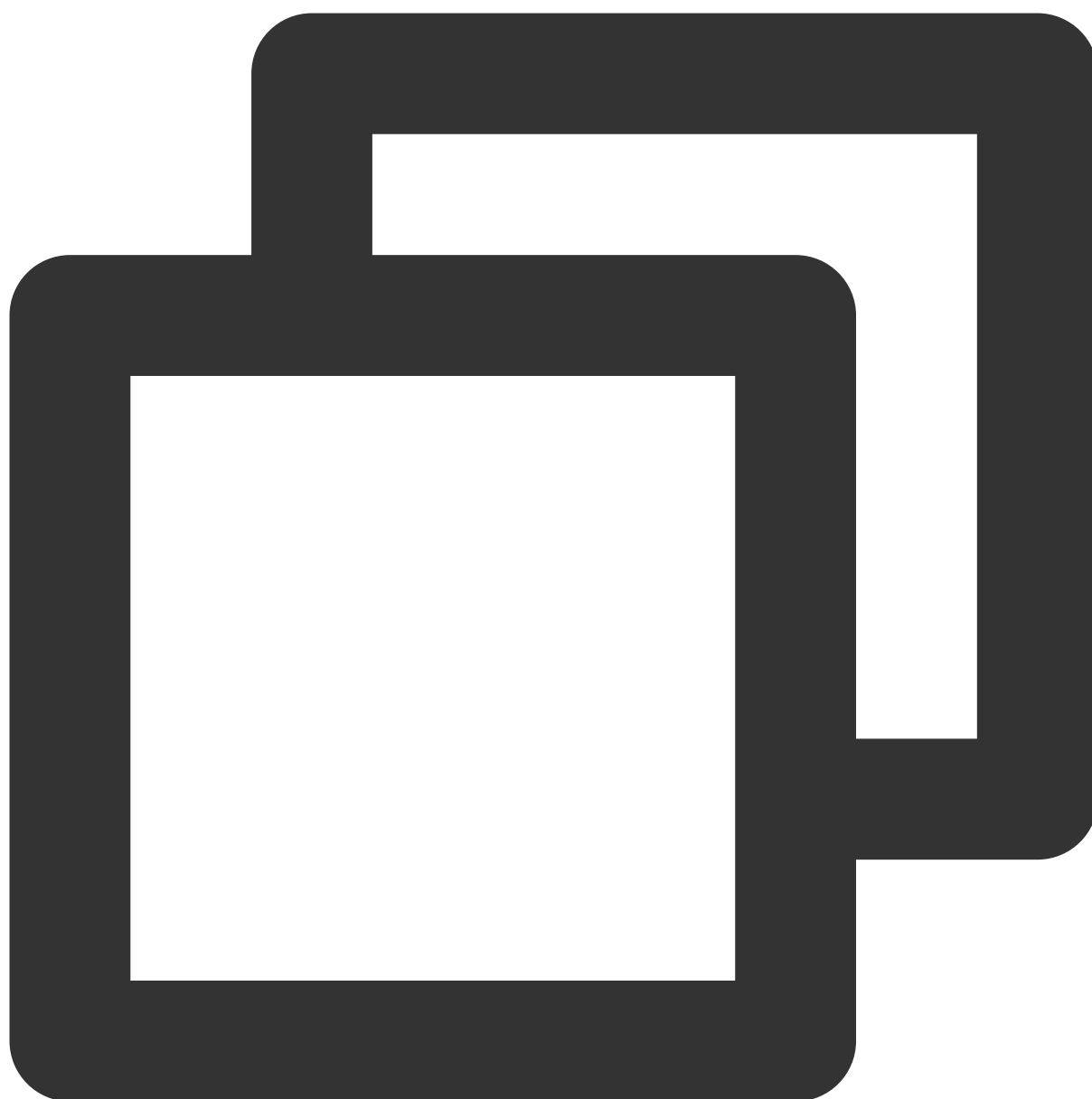


```
TO_TIMESTAMP('2007T') -- `2007T`  
TO_TIMESTAMP('2007-02-23T12:14:33.079-08:00') -- `2007-02-23T12:14:33.079-08:00`
```

UTCNOW

The UTCNOW function returns the current timestamp in UTC.

Syntax



UTCNOW ()

Sample



```
UTCNOW() -- 2019-01-01T14:23:12.123Z
```

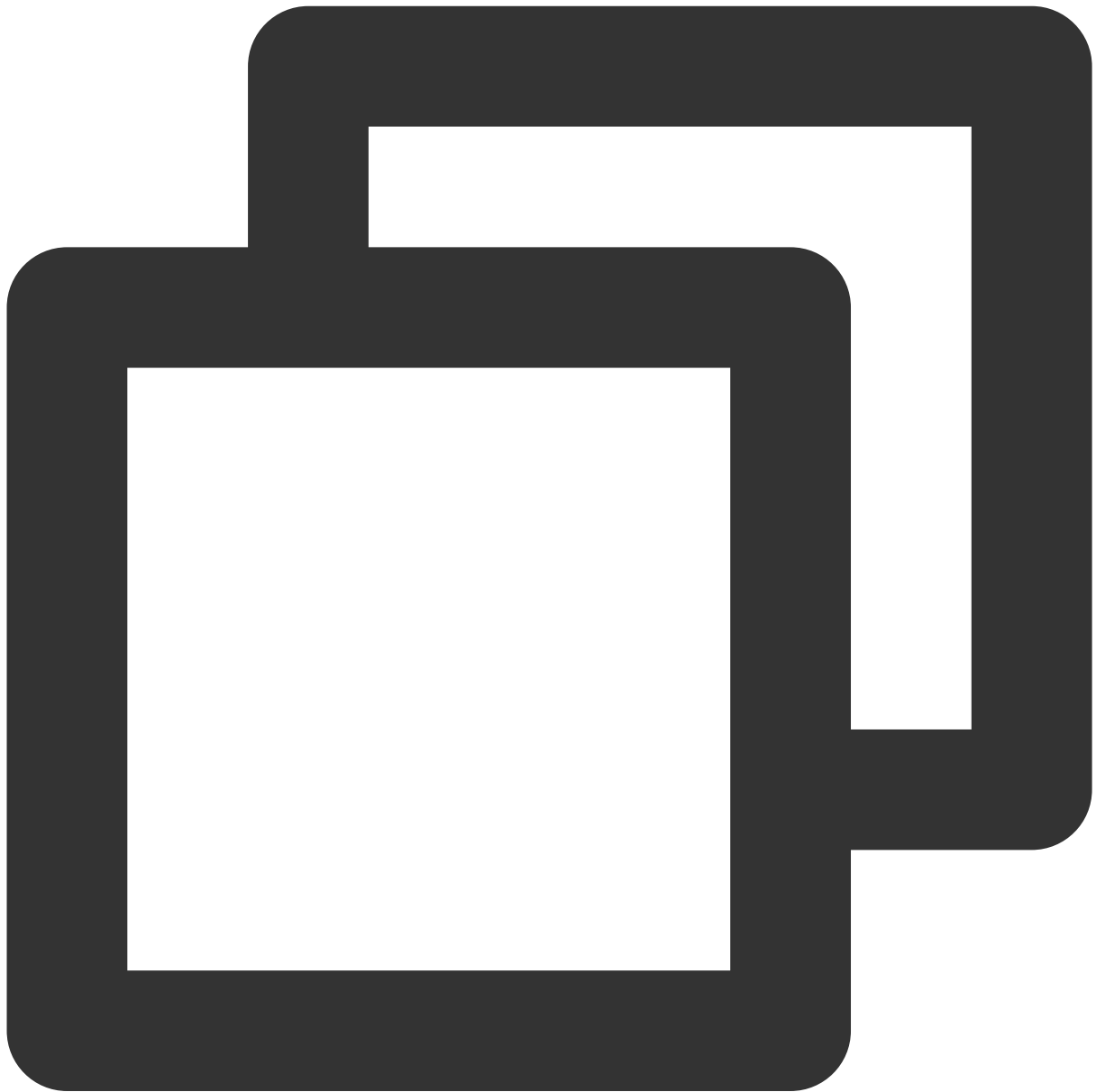
String Functions

COS Select supports the following string functions:

CHAR_LENGTH, CHARACTER_LENGTH

Both `CHAR_LENGTH` and `CHARACTER_LENGTH` can compute the number of characters in a string, and they have the same semantics.

Syntax



```
CHAR_LENGTH ( string )
```

Note:

The `string` parameter specifies the string for character counting

Sample



```
CHAR_LENGTH('null')      -- 4
CHAR_LENGTH('tencent')    -- 7
```

LOWER

The LOWER function converts all uppercase letters in the specified string to lowercase letters, with all non-uppercase letters left unchanged.

Syntax



```
LOWER ( string )
```

Note:

The `string` parameter specifies the string for which to convert uppercase letters to lowercase letters.

Sample



```
LOWER('TENcent') -- 'tencent'
```

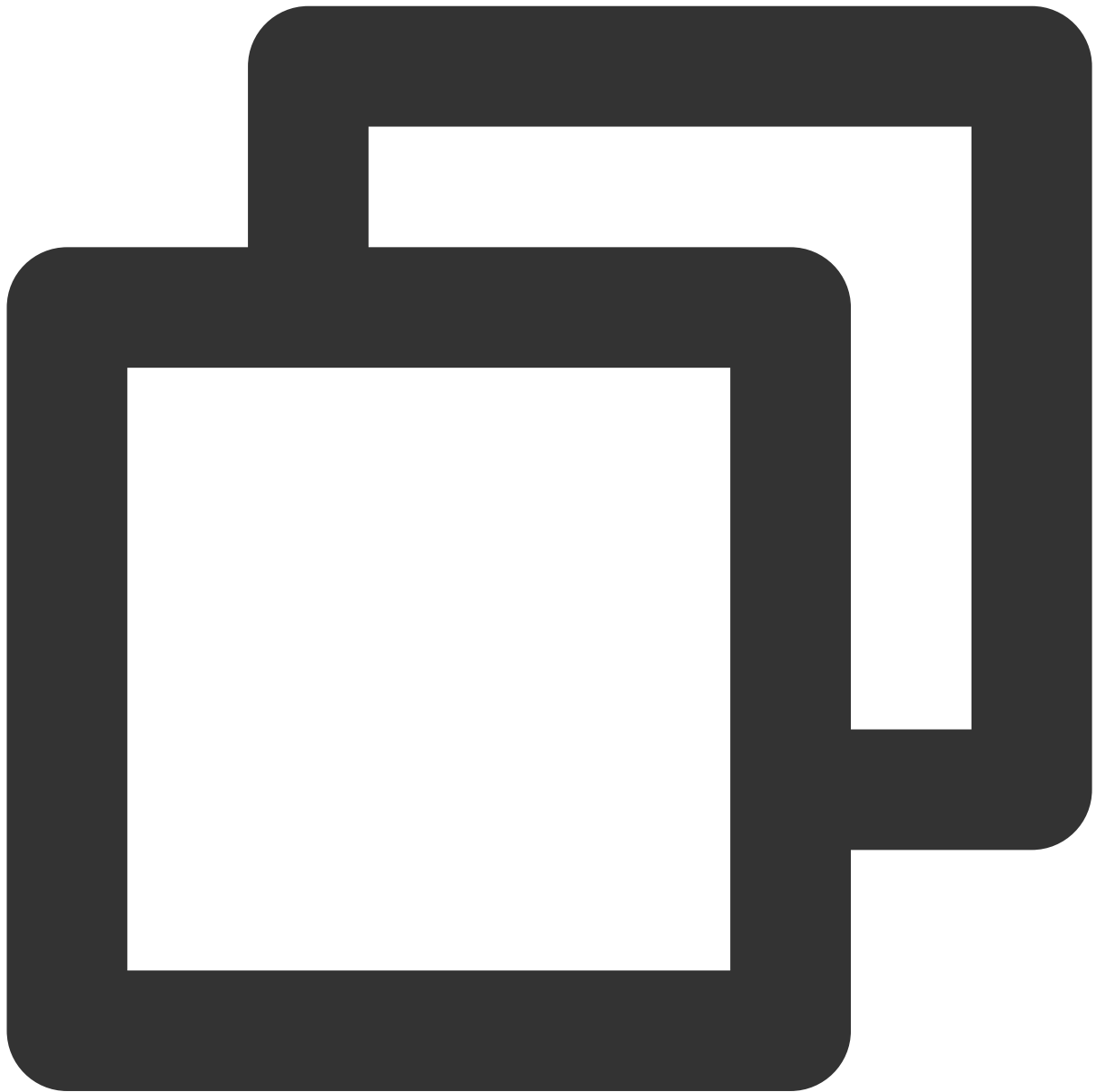
SUBSTRING

The SUBSTRING function returns a substring of a string. You can specify an index from which the SUBSTRING function will extract the remainder of the original string based on the length of the specified substring and return the result.

Note:

If the input string contains only 1 character, and the index is set to be greater than 1, the SUBSTRING function will automatically switch it to 1.

Syntax



```
SUBSTRING( string FROM start [ FOR length ] )
```

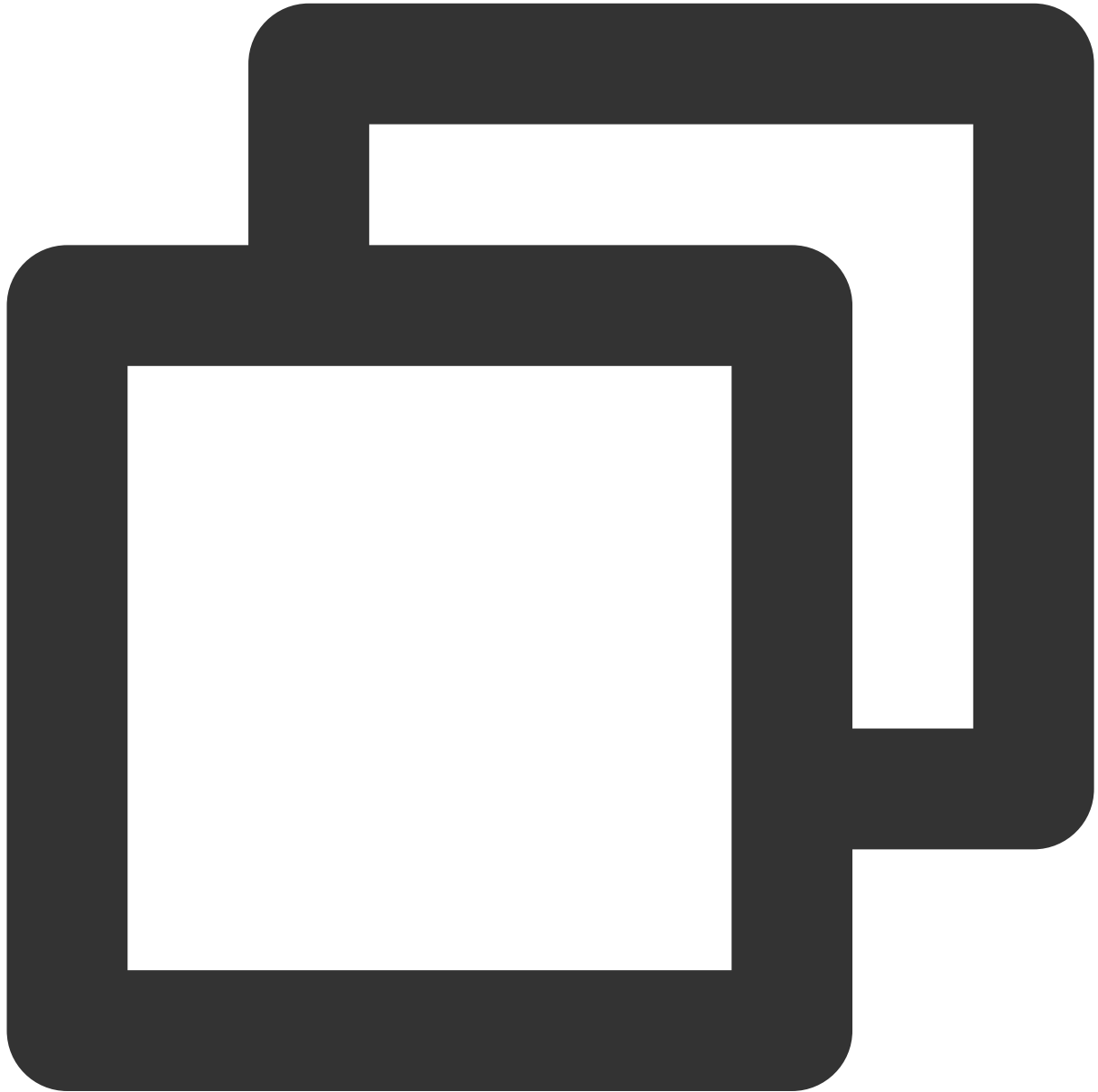
Note:

The `string` parameter specifies the string from which to extract a substring.

The `start` parameter represents an index value of the string as the starting position for extraction.

The `length` parameter specifies the length of the substring. If the length of the substring is not specified, the remainder of the string will be extracted.

Sample

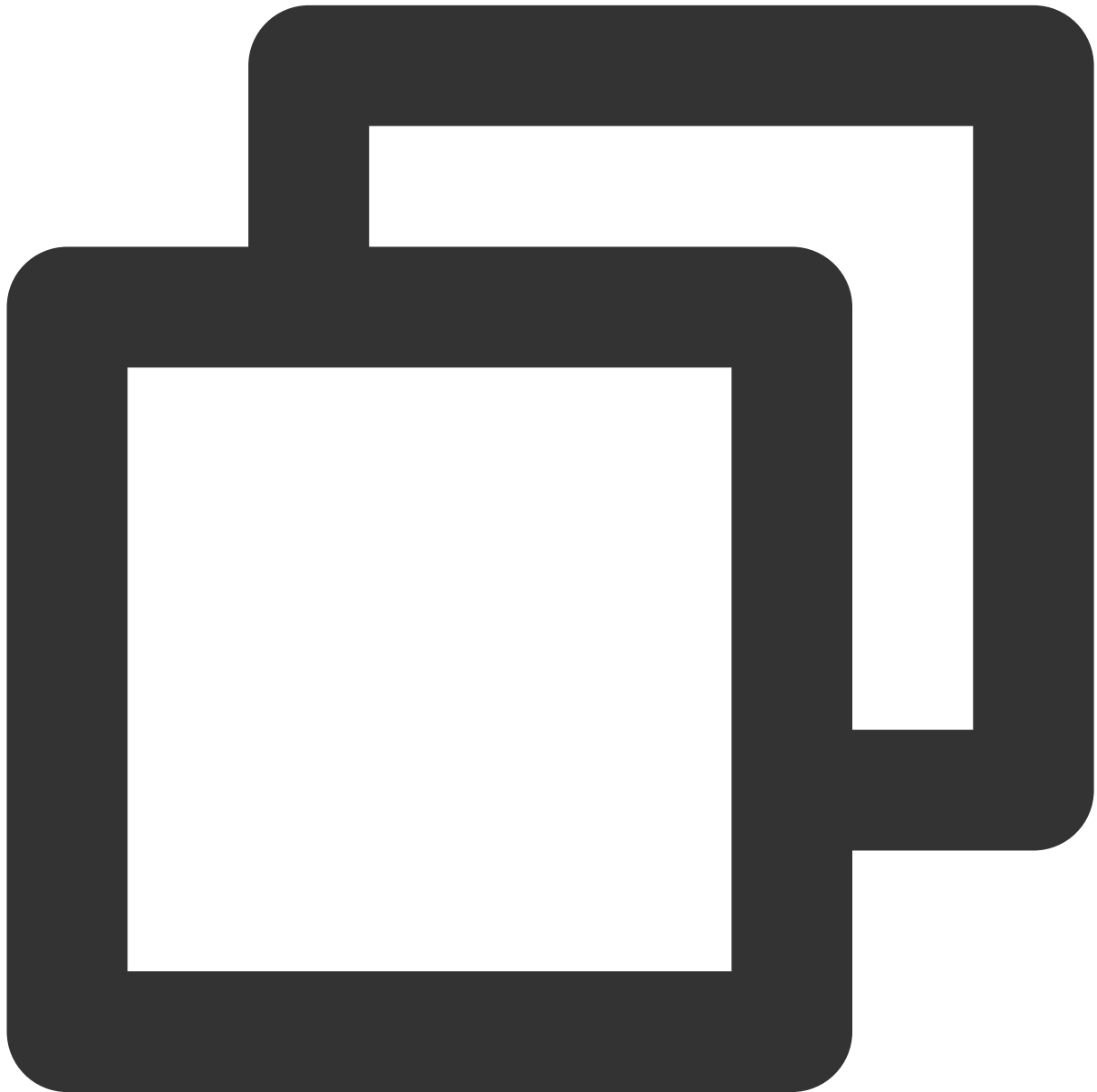


```
SUBSTRING("123456789", 0)      -- "123456789"
SUBSTRING("123456789", 1)      -- "123456789"
SUBSTRING("123456789", 2)      -- "23456789"
SUBSTRING("123456789", -4)     -- "123456789"
SUBSTRING("123456789", 0, 999) -- "123456789"
SUBSTRING("123456789", 1, 5)   -- "12345"
```


TRIM

The TRIM function deletes all characters before the first character or after the last character of the specified string. " " is the default character to be deleted.

Syntax



```
TRIM ( [[LEADING | TRAILING | BOTH remove_chars] FROM] string )
```

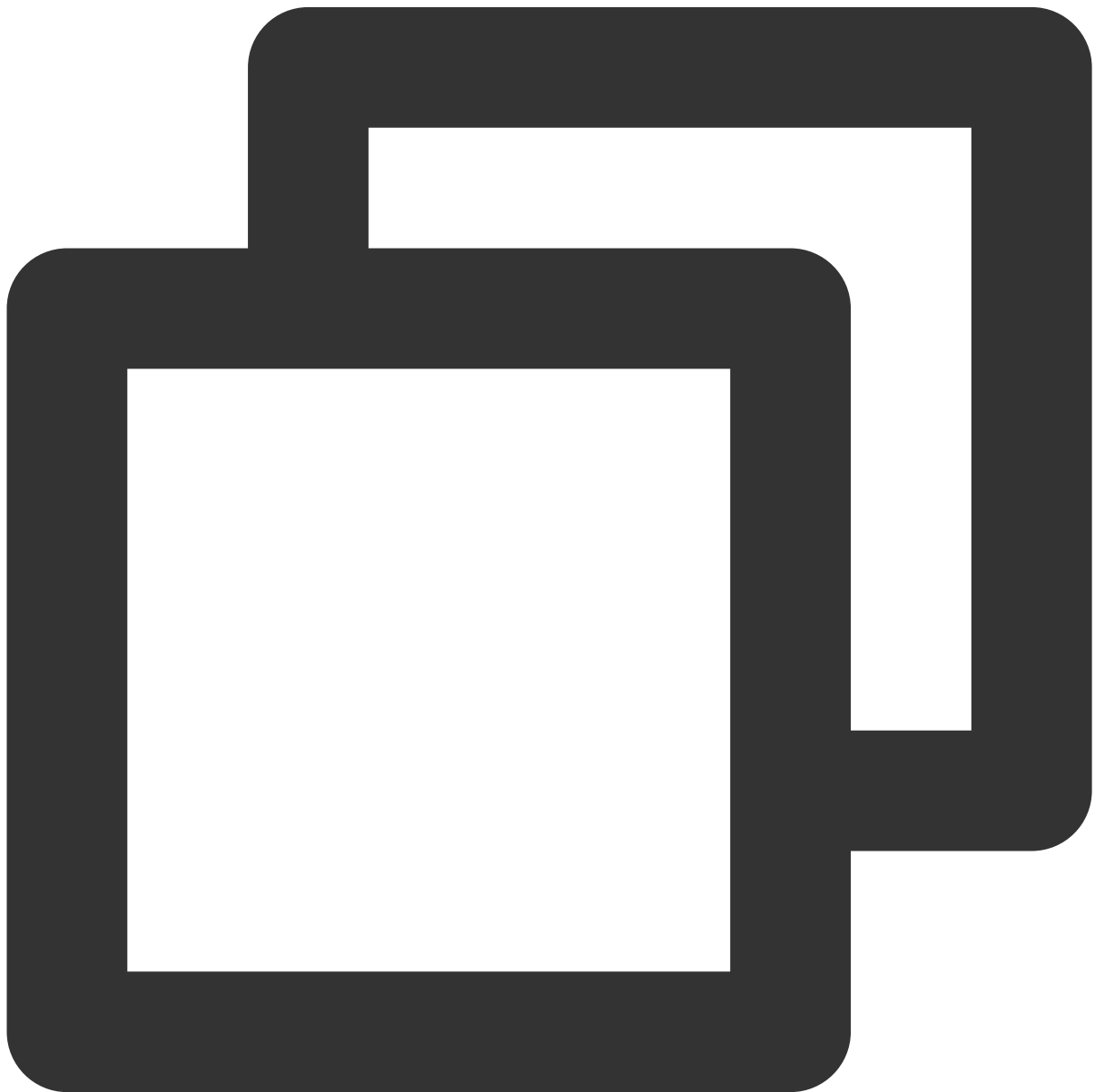
Note:

The `string` parameter specifies the string to be manipulated.

The `LEADING | TRAILING | BOTH` parameter specifies the extra characters to be deleted, which can be before the string (LEADING), after the string (TRAILING), or both (BOTH).

The `remove_chars` parameter specifies the type of extra characters to be deleted. It can be a string containing more than one characters. The TRIM function will delete all extra characters of the corresponding type that are identified by the TRIM function before or after the `string` parameter.

Sample



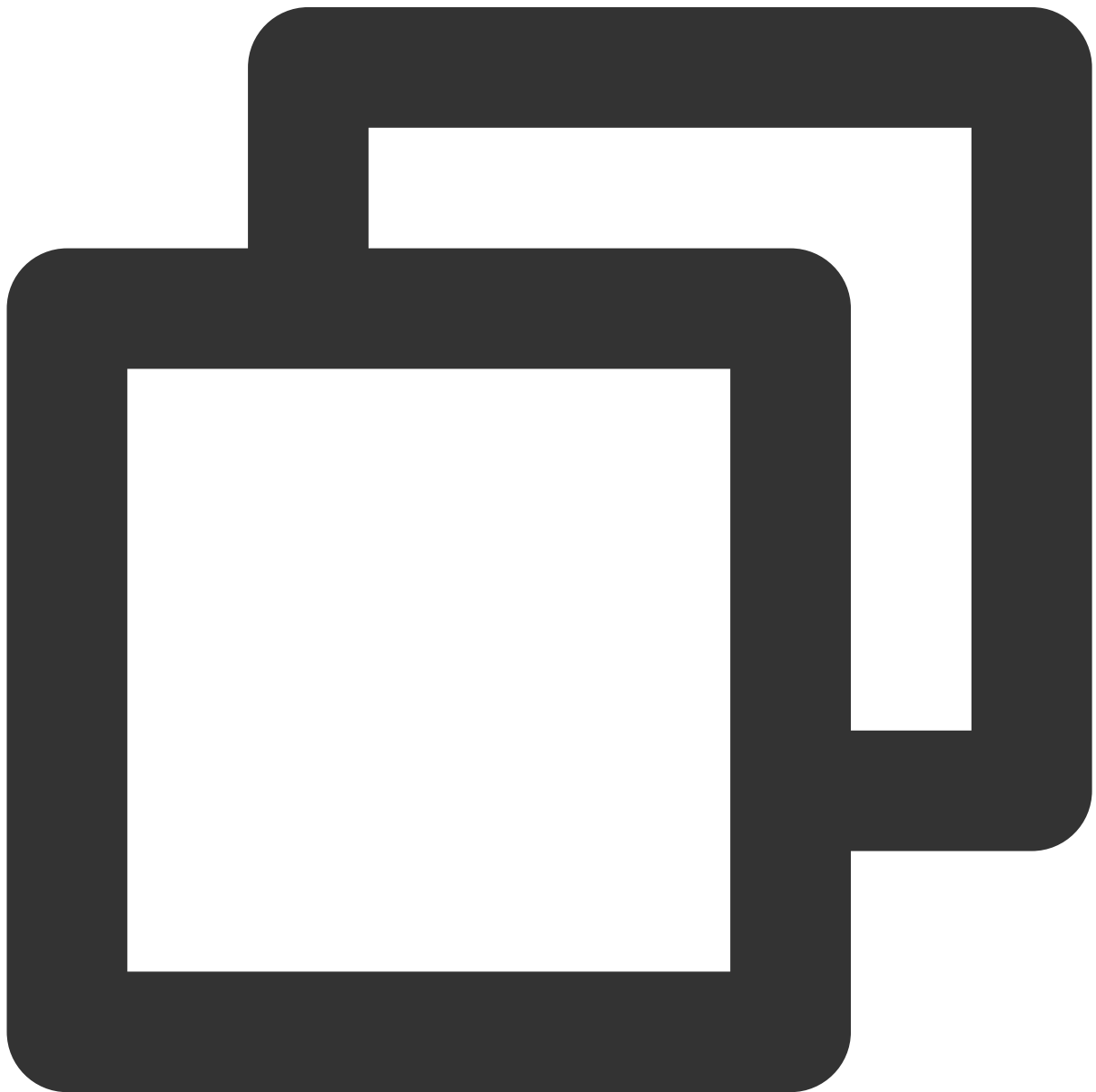
```
TRIM('      foobar      ')      -- 'foobar'
```

```
TRIM('    \\tfoobar\\t    ')      -- '\\tfoobar\\t'  
TRIM(LEADING FROM '    foobar    ') -- 'foobar'  
TRIM(TRAILING FROM '    foobar    ') -- 'foobar'  
TRIM(BOTH FROM '    foobar    ')   -- 'foobar'  
TRIM(BOTH '12' FROM '1112211foobar22211122') -- 'foobar'
```

UPPER

The UPPER function converts all lowercase letters to uppercase letters with non-lowercase letters left unchanged.

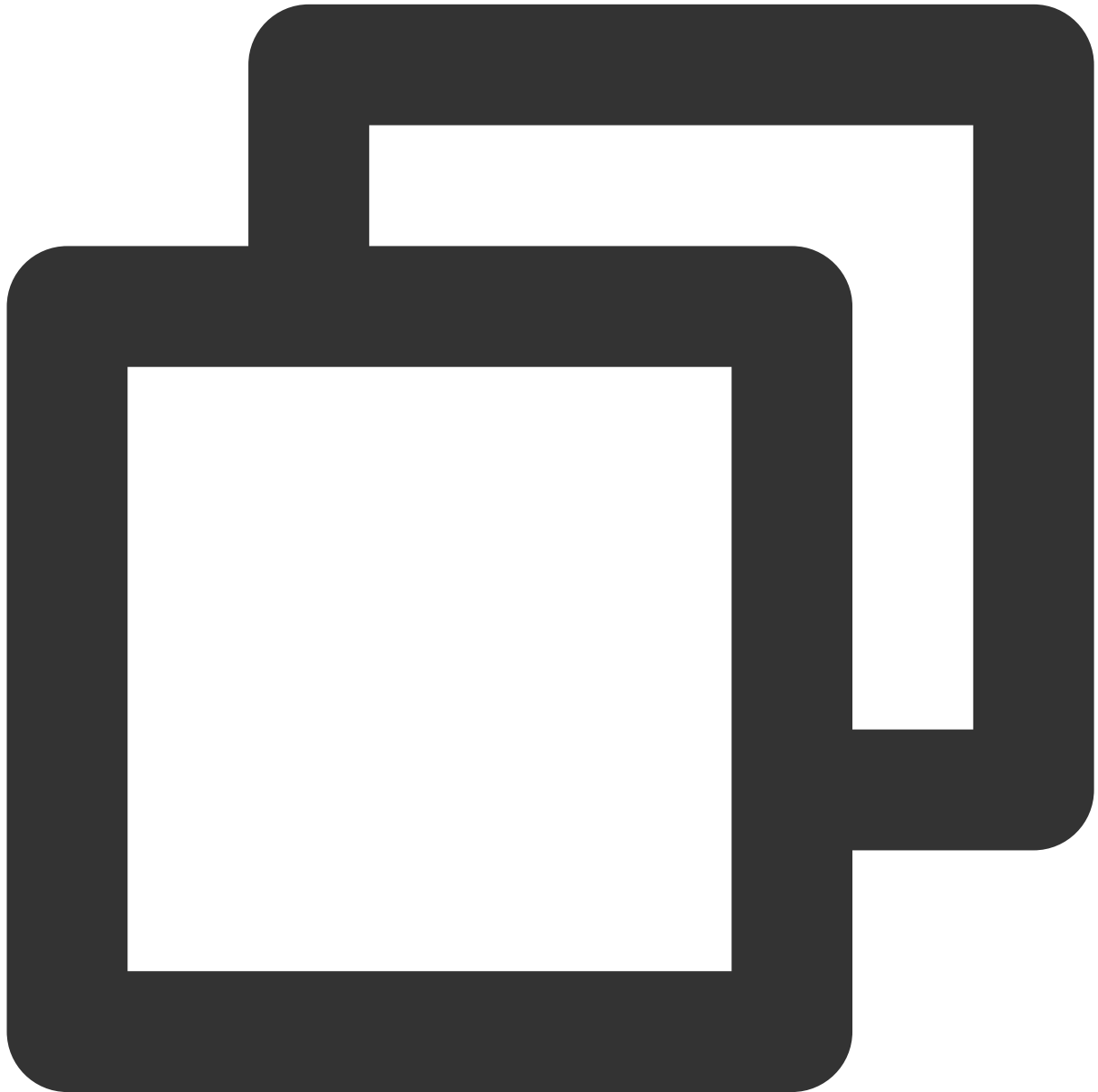
Syntax



```
UPPER ( string )
```

Note:

The `string` parameter specifies the string to be converted to uppercase letters.

Sample

```
UPPER('tenCENT') -- 'TENCENT'
```

Reserved Words

Last updated : 2024-03-25 15:28:24

In order to ensure normal operation and subsequent expansion, the COS Select feature has the following reserved fields such as function name, data type, and operator. You may use these fields to facilitate your SQL queries.

No.	Field	No.	Field	No.	Field	No.	Field	No.
1	absolute	51	create	101	goto	151	octet_length	201
2	action	52	cross	102	grant	152	of	202
3	add	53	current	103	group	153	on	203
4	all	54	current_date	104	having	154	only	204
5	allocate	55	current_time	105	hour	155	open	205
6	alter	56	current_timestamp	106	identity	156	option	206
7	and	57	current_user	107	immediate	157	or	207
8	any	58	cursor	108	in	158	order	208
9	are	59	date	109	indicator	159	outer	209
10	as	60	day	110	initially	160	output	210
11	asc	61	deallocate	111	inner	161	overlaps	211
12	assertion	62	dec	112	input	162	pad	212
13	at	63	decimal	113	insensitive	163	partial	213
14	authorization	64	declare	114	insert	164	pivot	214
15	avg	65	default	115	int	165	position	215
16	bag	66	deferrable	116	integer	166	precision	216
17	begin	67	deferred	117	intersect	167	prepare	217
18	between	68	delete	118	interval	168	preserve	218
19	bit	69	desc	119	into	169	primary	219
20	bit_length	70	describe	120	is	170	prior	220

21	blob	71	descriptor	121	isolation	171	privileges	221
22	bool	72	diagnostics	122	join	172	procedure	222
23	boolean	73	disconnect	123	key	173	public	223
24	both	74	distinct	124	language	174	read	224
25	by	75	domain	125	last	175	real	225
26	cascade	76	double	126	leading	176	references	226
27	cascaded	77	drop	127	left	177	relative	227
28	case	78	else	128	level	178	restrict	228
29	cast	79	end	129	like	179	revoke	229
30	catalog	80	end-exec	130	limit	180	right	230
31	char	81	escape	131	list	181	rollback	231
32	char_length	82	except	132	local	182	rows	232
33	character	83	exception	133	lower	183	schema	233
34	character_length	84	exec	134	match	184	scroll	234
35	check	85	execute	135	max	185	second	235
36	clob	86	exists	136	min	186	section	236
37	close	87	external	137	minute	187	select	237
38	coalesce	88	extract	138	missing	188	session	238
39	collate	89	false	139	module	189	session_user	239
40	collation	90	fetch	140	month	190	set	240
41	column	91	first	141	names	191	sexp	241
42	commit	92	float	142	national	192	size	242
43	connect	93	for	143	natural	193	smallint	-
44	connection	94	foreign	144	nchar	194	some	-
45	constraint	95	found	145	next	195	space	-

46	constraints	96	from	146	no	196	sql	-
47	continue	97	full	147	not	197	sqlcode	-
48	convert	98	get	148	null	198	sqlerror	-
49	corresponding	99	global	149	nullif	199	sqlstate	-
50	count	100	go	150	numeric	200	string	-

Data Types

Last updated : 2024-03-25 15:28:24

Overview

COS Select supports a wide variety of primitive data types.

The data type directly supported by a compiler is called a **primitive data type**.

Data Type Conversion

COS Select uses the CAST function to determine the data type of your input data. In general, if you do not specify a data type through the CAST function, COS Select will treat the input data as string type.

For more information on the CAST function, see the [CAST](#) section in the SQL function documentation.

Supported Data Types

COS Select supports the following primitive data types:

Name	Description	Example
bool	TRUE/FALSE	FALSE
int, integer	An 8-byte signed integer Value range: -9,223,372,036,854,775,808 - 9,223,372,036,854,775,807	100000
string	A UTF-8-encoded string with a character length in the range of 1 - 2,147,483,647	'xyz'
float	An 8-byte floating point	CAST(0.456 AS FLOAT)
decimal, numeric	A decimal value with a maximum precision of 38 decimal places and in the range of -2^{31} - $2^{31}-1$	123.456
timestamp	A timestamp represents a certain moment and can be expressed with any precision. A timestamp in text format follows the W3C specification but needs to end with "T" (unless recorded in days).	CAST('2007-04-05T14:30Z' AS TIMESTAMP)

<p>When a fractional second is used, it should be accurate to at least one decimal place (with no upper limit on the number of decimal places). The offset of local time can be expressed by the offset expressed by hours and minutes when compared to UTC, or "Z" can be used to express the offset of local time from UTC. The time offset does not need to be displayed when only the date is recorded.</p>	
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

Operators

Last updated : 2024-03-25 15:28:24

COS Select supports the following operators:

Operator Type	Operator
Logical operator	AND, NOT, OR
Comparison operator	<, >, <=, >=, =, <>, !=, BETWEEN, IN, such as <code>IN ('a', 'b', 'c')</code>
Pattern matching operator	LIKE
Mathematical operator	+, -, *, %

Priority of Operators

The following table shows the priority of operators in descending order:

Operator/Element	Association	Use Case
-	Right	Unary subtraction
*, /, %	Left	Multiplication, division, and modulo
+, -	Left	Addition and subtraction
IN	-	Setting membership
BETWEEN	-	In the range of ()
LIKE	-	String pattern matching
<>	-	Less than and greater than
=	Right	Equivalence and assignment
NOT	Right	Logical NOT
AND	Left	Logical AND
OR	Left	Logical OR

Log Management

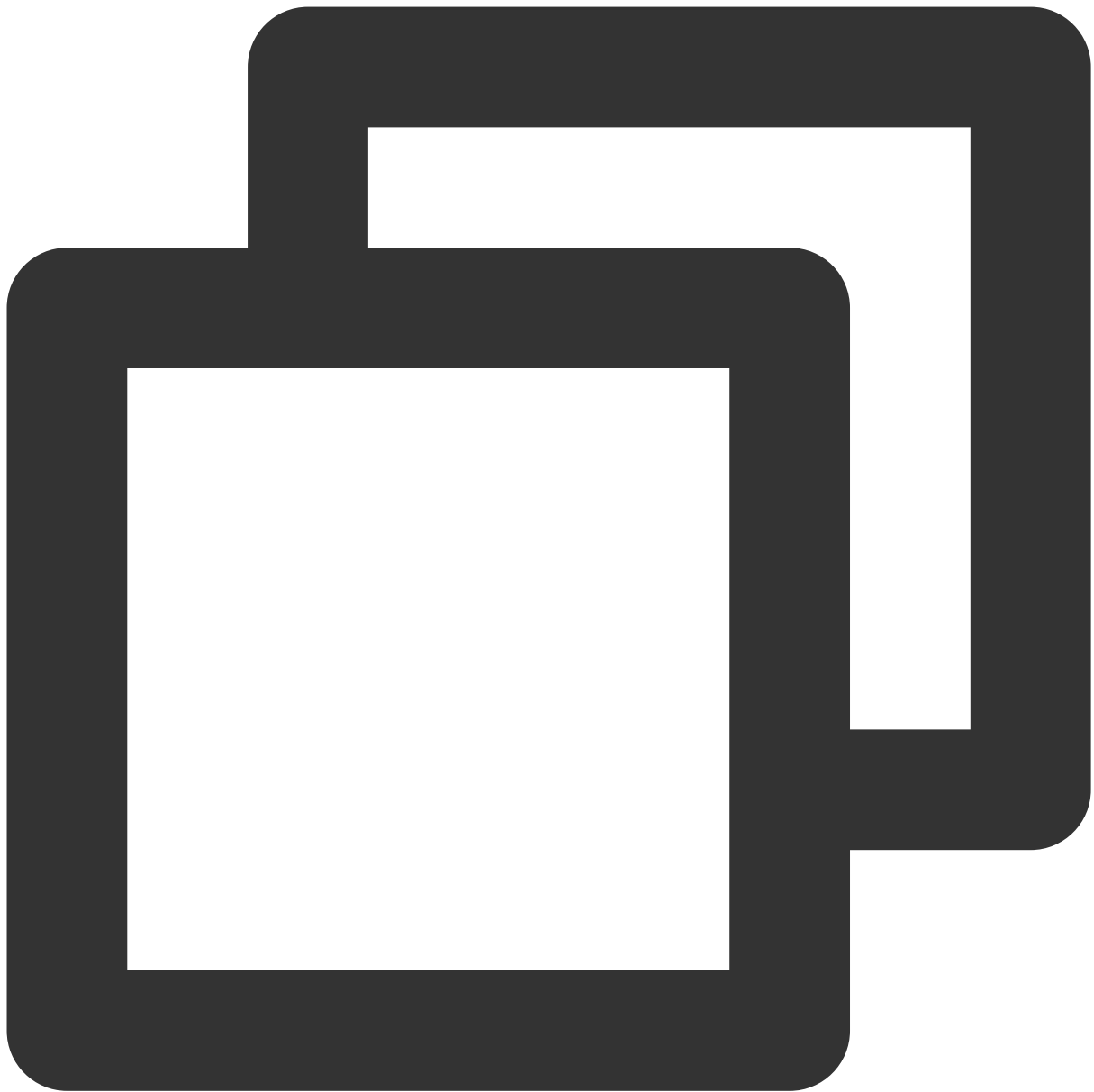
Logging Overview

Last updated : 2024-03-25 15:28:24

Overview

The logging feature allows you to record the access information of a source bucket and save it to a destination bucket as logs.

In the destination bucket, the log path is:



```
destination bucket/path prefix{YYYY}/{MM}/{DD}/{time}_{random}_{index}
```

Logs are generated every 5 minutes (one record per line). Each record contains multiple fields, and fields are separated by a space. The file size for a log file is up to 256 MB. If the log file reaches the file size limit in 5 minutes, a new log file will be created. The logging fields supported are as follows:

No.	Field	Description	Example
1	eventVersion	Log version	1.0
2	bucketName	Bucket name	examplebucket-1250000000

3	qcsRegion	Request region	ap-beijing
4	eventTime	Event time (request end time, which is a timestamp in UTC+0 time zone)	2018-12-01T11:02:33Z
5	eventSource	Access domain name	examplebucket-1250000000.cos.ap- guangzhou.myqcloud.com
6	eventName	Event name	UploadPart
7	remotelp	Source IP	192.168.0.1
8	userSecretKeyId	User access KeyId	AKIDNYVCdoJQyGJ5b1234
9	reservedFiled	Reserved field	Displayed as –
10	reqBytesSent	Request bytes	83886080
11	deltaDataSize	Change in storage made by the request (in bytes)	808
12	reqPath	Requested file path	/folder/text.txt
13	reqMethod	Request method	put
14	userAgent	User UA	cos-go-sdk-v5.2.9
15	resHttpCode	HTTP response status code	404
16	resErrorCode	Error code	NoSuchKey
17	resErrorMsg	Error message	The specified key does not exist.
18	resBytesSent	Bytes returned	197
19	resTotalTime	Total time used by the request (in milliseconds, i.e., the time between	4295

		the last byte of the response and the first byte of the request)	
20	logSourceType	Source type of the log	<code>USER</code> (user access requests), <code>CDN</code> (CDN origin-pull requests)
21	storageClass	Storage class	STANDARD, STANDARD_IA, ARCHIVE
22	accountId	Bucket owner ID	1000000000001
23	resTurnAroundTime	Time used by the request server (in milliseconds, i.e., the time between the first byte of the response and the last byte of the request)	4295
24	requester	Requester account. If the requester is a root account, this parameter is in the format of <code>root account UIN:root account UIN</code> . If the requester is a sub-account, this parameter is in the format of <code>root account UIN:sub-account UIN</code> . If the requester is a service account , this parameter is in the format of <code>authorized root account UIN:role ID</code> .	1000000000001:1000000000001

		If the requester is anonymous, this parameter is displayed as <code>-</code> .	
25	requestId	Request ID	NWQ1ZjY4MTBfMjZiMjU4NjRfOWI1N180NDBiYTY=
26	objectSize	Object size, in bytes	808. If you use multipart upload, <code>objectSize</code> will only be displayed when the upload is complete, and will be <code>-</code> during the multipart upload process
27	versionId	Object version ID	Random string
28	targetStorageClass	Destination storage class, recorded for replication requests	STANDARD, STANDARD_IA, ARCHIVE
29	referer	HTTP referer of the request	<code>*.example.com</code> or <code>111.111.111.1</code>
30	requestUri	Request URI	"GET /fdgfdgsf%20/%E6%B5%AE%E7%82%B9%E6%95%B0 HTTP/1.1"
31	vpclId	VPC request ID	"0" (non-VPC)/"12345" (VPC, a non-"0" string)

Note:

Currently, COS offers the logging feature only in the Beijing, Shanghai, Guangzhou, Nanjing, Chongqing, Chengdu, Hong Kong (China), Singapore, Seoul, Toronto, Silicon Valley, and Mumbai regions.

The destination bucket and source bucket must reside in the same region.

The destination bucket that stores logs can be the source bucket itself (not recommended).

Currently, logs will be generated only when the bucket is accessed through XML APIs and XML API-based SDKs or tools, not via JSON APIs or JSON API-based SDKs or tools.

Depending on customer needs and business development, COS may add new fields to the access logs. Be sure to prepare for this when parsing the logs.

Enabling Logging

Using the Console

You can quickly enable logging in the console. For detailed directions, see [Setting Logging](#).

Using APIs

To enable logging for a bucket using APIs, follow the steps below:

1. Create a log role.
2. Grant the log role permissions.
3. Enable logging.

1. Create a log role

Create a log role. For more information, see [CreateRole](#).

Here, `roleName` must be `CLS_QcsRole` .

`policyDocument` must be:



```
{
  "version": "2.0",
  "statement": [{
    "action": "name/sts:AssumeRole",
    "effect": "allow",
    "principal": {
      "service": "cls.cloud.tencent.com"
    }
  }]
}
```

2. Grant the log role permissions

Grant the log role permissions. For more information about the API, see [AttachRolePolicy](#).

Here, `policyName` is set to `QcloudCOSAccessForCLSRole`, `roleName` is the `CLS_QcsRole` in step 1, or the `roleID` returned when `roleName` was created.

3. Enable logging

Call the API to enable logging. For more information about the API, see [PUT Bucket logging](#). Note that the destination bucket to store the logs should be in the same region as the source bucket.

Restrictions on Log Management

Last updated : 2024-03-25 15:28:24

The log management feature is used to record the detailed access information of the specified source bucket and store the information in the specified bucket in the form of log files to facilitate bucket management.

Currently, the use limits of the log management feature are as follows:

Delivery frequency limit: A log is generated every 5 minutes.

Limit on the size of log file to be delivered: The log file to be delivered each time can be up to 256 MB. After this limit is exceeded, a new file will be delivered.

Format of the log to be delivered: Each record is saved as one line and can contain multiple fields separated by space.

Restrictions on the fields to be delivered: For more information on the fields to be delivered, see [Log Management Overview](#).

Invalid fields: If a field in a log contains the `-` character, the field is an invalid record or the default record.

Content Recorded in a Log

The requests initiated by you to upload, download, or delete an object, to create or delete a bucket, or to modify the bucket configuration.

The requests generated by content delivery through CDN and data pull from the COS origin server.

Content Not Recorded in a Log

Offline origin-pull requests: After origin-pull is configured, if there are no objects in COS, the data will be downloaded from the origin server specified by you. This download operation is an offline origin-pull request and will not be recorded in the log.

Redirects within a static website: If you configure redirection in the static website feature, you may be redirected to another page when accessing index.html. Such redirects will not be recorded in the log.

Lifecycle transitions and deletions: If you configure the lifecycle to transition or delete objects after expiration, the operations of transitioning or deleting objects will be performed by the COS backend and will not be recorded in the log.

Operations of listing objects and uploading inventory reports: The inventory feature automatically lists all or specified objects in your buckets and delivers the generated inventory report to your buckets on a periodical basis. The operations of listing objects and delivering inventory reports will not be recorded in the log.

Operations of cross-region object replication: Cross-region replication involves getting an object from the source bucket and uploading it to the destination bucket. These operations are performed by the COS backend and will not

be recorded in the log.

Object downloads in the COS Select feature: The COS Select feature allows you to extract an object which needs to be obtained from a storage device first. Object downloads are performed by the COS backend and will not be recorded in the log.

Using COS to Store Tencent Cloud Product Logs

Last updated : 2024-03-25 15:28:24

Overview

The process of using Tencent Cloud products generates a massive number of logs, which can be used to help you analyze and make decisions for your business. For your convenience, you can store these logs persistently in COS, and then easily access them for analysis by using various APIs, SDKs or tools.

COS log storage offers the following benefits:

Persistent storage: COS provides stable, persistent storage that allows you to store logs in COS at a very low cost. You can access your logs anytime, anywhere for business analysis or decision-making.

COS Select: This feature allows for the simple extraction of logs stored on COS. You can combine log fields to make it easier to extract the information you need and reduce data download traffic.

Data analysis: You can analyze one or more COS logs using Sparkling, which then can be used to facilitate and provide a basis for important business decisions.

Shipping Logs

You can ship your Tencent Cloud product logs to COS in one of the following two ways:

Use the Tencent Cloud log shipping feature in which logs are directly shipped to COS. Products such as COS and CloudAudit (CA) support directly shipping logs to COS.

Use the CLS log shipping feature in which logs are first shipped to CLS and then transferred to COS for persistent storage.

The table below shows the particular log shipping method or methods currently supported by various Tencent Cloud products:

Product	Supports Direct Shipping to COS	Supports Shipping to CLS
CA	Yes	No
CLB	No	Yes
CKafka	Yes	No
API Gateway	No	Yes

SCF	No	Yes
TKE	No	Yes
CSS	No	Yes
TCB	No	Yes (however, TCB does not support shipping logs to COS through CLS)
COS	Yes	Yes for accounts on the allowlist. You can contact us to add your account to the allowlist.

Directly shipping logs to COS

The Tencent Cloud products outlined below allow you to ship logs directly to COS by setting log shipping rules as instructed in the product-specific documentation.

Product	Log Shipping Document	Log Shipping Interval	Log Shipping Path
CA	Click here to view	10-15 min	clouddaudit/customprefix/timestamp
CKafka	Click here to view	5-60 min You can specify the interval	instance id/topic id/timestamp
COS	Click here to view	5 min	You can specify the path prefix. We recommend specifying an identifiable path, e.g. <code>cos_bucketname_access_log/timestamp</code> .

Note:

CKafka supports shipping the message data that it generates. However, to get logs on certain actions such as the creation of CKafka instances, you need to ship the CA logs.

Using CLS to ship logs to COS

The Tencent Cloud products outlined below allow you to ship logs directly to CLS for search and analysis. With CLS, you can further ship the logs to COS for persistent storage, thus reducing your costs and allowing for convenient offline analysis.

Product	Documentation
API Gateway	Click here to view

TKE	Click here to view
CSS	Click here to view

CLS can ship the following three types of logs to COS:

CSV-formatted logs: Logs formatted with comma-separated values. For more information, see [Shipping CSV-Formatted Logs](#).

JSON-formatted logs: Logs in JSON format. For more information, see [Shipping JSON-Formatted Logs](#).

Original logs: Logs in their original format, including those with full text in a single line, full text in multi-lines, and comma-separated values (for certain products). For more information, see [Shipping Original Logs](#).

To ship logs from CLS to COS, you need to do the following:

1. Choose a Tencent Cloud product that meets your business needs and configure a logset and log topic as instructed in the log shipping documentation provided above to connect your business data to CLS.
2. With your particular business needs in mind, select the most suitable log type to ship to COS. We recommend that you use the product name as the path prefix to differentiate product logs, for example, you could use the following name for TKE logs: `TKE_tkeid_log/timestamp`.
3. Once the shipping rule is configured, you can configure COS event notifications for file uploads through SCF. You can perform further operations on logs shipped to COS based on these event notifications. For more information, see [Event Notifications](#).

Log Analysis

Downloading logs locally for offline analysis

You can download logs to the local file system through the Tencent Cloud console, or by using various SDKs, APIs, and tools. To do so, follow the instructions in one of the following documents and replace each file path in the code with your log storage path:

Download Method	Document
Console	Click here to view
COSBrowser	Click here to view
COSCMD	Click here to view
Android SDK	Click here to view
C SDK	Click here to view
C++ SDK	Click here to view

.NET SDK	Click here to view
Go SDK	Click here to view
iOS SDK	Click here to view
Java SDK	Click here to view
JavaScript SDK	Click here to view
Node.js SDK	Click here to view
PHP SDK	Click here to view
Python SDK	Click here to view
WeChat Mini Program SDK	Click here to view
API	Click here to view

Analyzing logs using COS Select

You have the option to extract and analyze COS logs stored in CSV or JSON format directly by using the COS Select feature. It enables you to query desired log fields, which largely reduces the number of logs transferred by COS, thus lowering usage costs and improving data access speed. For more information on COS Select, see [Select Overview](#).

Currently, you can access COS Select using the console or APIs.

Method	Instructions
Console	Click here to view
API	Click here to view

Data Disaster Recovery

Versioning

Overview

Last updated : 2024-03-25 15:33:39

Overview

By enabling versioning, you can store multiple versions of an object in the same bucket. For example, you can store multiple objects with the same object key "picture.jpg" in one bucket, but the objects have different version IDs such as 100000, 100101, and 120002. You can query, delete, or restore the objects by version ID. Versioning enables you to recover from data loss caused by accidental deletion or application failure. Versioning is ideal for the following scenarios:

If you need to delete objects (not permanently) in a versioning-enabled bucket, COS will insert delete markers for the deleted objects, and the markers will serve as the current object versions. You can restore an object to its previous version by the [delete marker](#).

COS will insert new version IDs for new objects you upload to replace existing ones, which you can restore by version ID.

Versioning States

There are three versioning states for a bucket:

Versioning not enabled: Bucket versioning is not enabled by default.

Versioning enabled: When bucket versioning is enabled, it will be applied to **all the objects** in the bucket. After versioning is enabled for the first time, objects uploaded to the bucket thereafter will be assigned a unique version ID.

Versioning suspended: Once the versioning state is changed from enabled to suspended (versioning cannot be disabled once enabled), objects uploaded to the bucket thereafter will no longer be subject to versioning.

Note:

1. Once versioning is enabled for the bucket, it cannot return to the prior status (initial status). However, you can suspend versioning for the bucket so that subsequent uploads of objects will not generate multiple versions.
2. For objects existing in the bucket before versioning is enabled, the version ID is "null".
3. Enabling or suspending versioning will change the way COS makes requests when handling the objects but not the objects themselves.
4. Only the root account and authorized sub-accounts can suspend versioning for a bucket.

Managing Objects in Buckets with Different Versioning States

You can upload, query, and delete objects in a bucket no matter what the versioning state is. When versioning is enabled or suspended, you can query and delete objects with or without version IDs.

If versioning is not enabled, you can upload, query, and delete objects in the same way as in an ordinary bucket. For more information, see [Object Management](#).

If versioning is enabled or suspended, you can specify version IDs while uploading, querying or deleting objects.

Delete markers will be added to deleted objects.

Managing objects in a versioning-enabled bucket

For objects in the bucket before versioning is enabled, their version ID will be “null”. Enabling versioning will change the way COS handles the objects, such as how COS makes requests, but not the objects themselves. Objects with the same name uploaded thereafter will be stored in the same bucket with different version IDs. You can manage objects in a bucket with versioning enabled as described below:

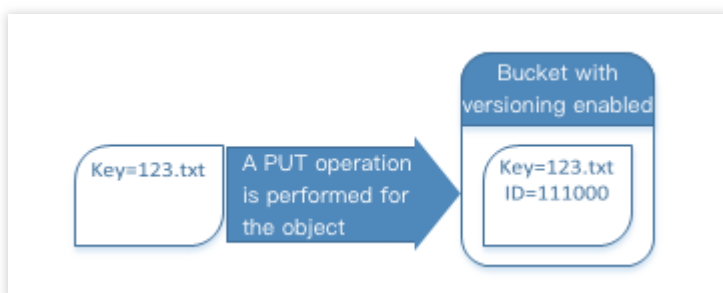
Note:

An object is uploaded to both buckets with versioning enabled and not enabled in the same way, but with different version IDs. If versioning is enabled, COS will assign a unique version ID to any object uploaded to the bucket; if versioning is not enabled, the version ID will remain null.

Uploading an object

If versioning is enabled for a bucket, when you perform PUT, POST, and COPY operations, COS will automatically assign a unique version ID to the objects added to the bucket.

As you can see below, when an object is uploaded to a bucket with versioning enabled, COS will assign it a unique version ID.



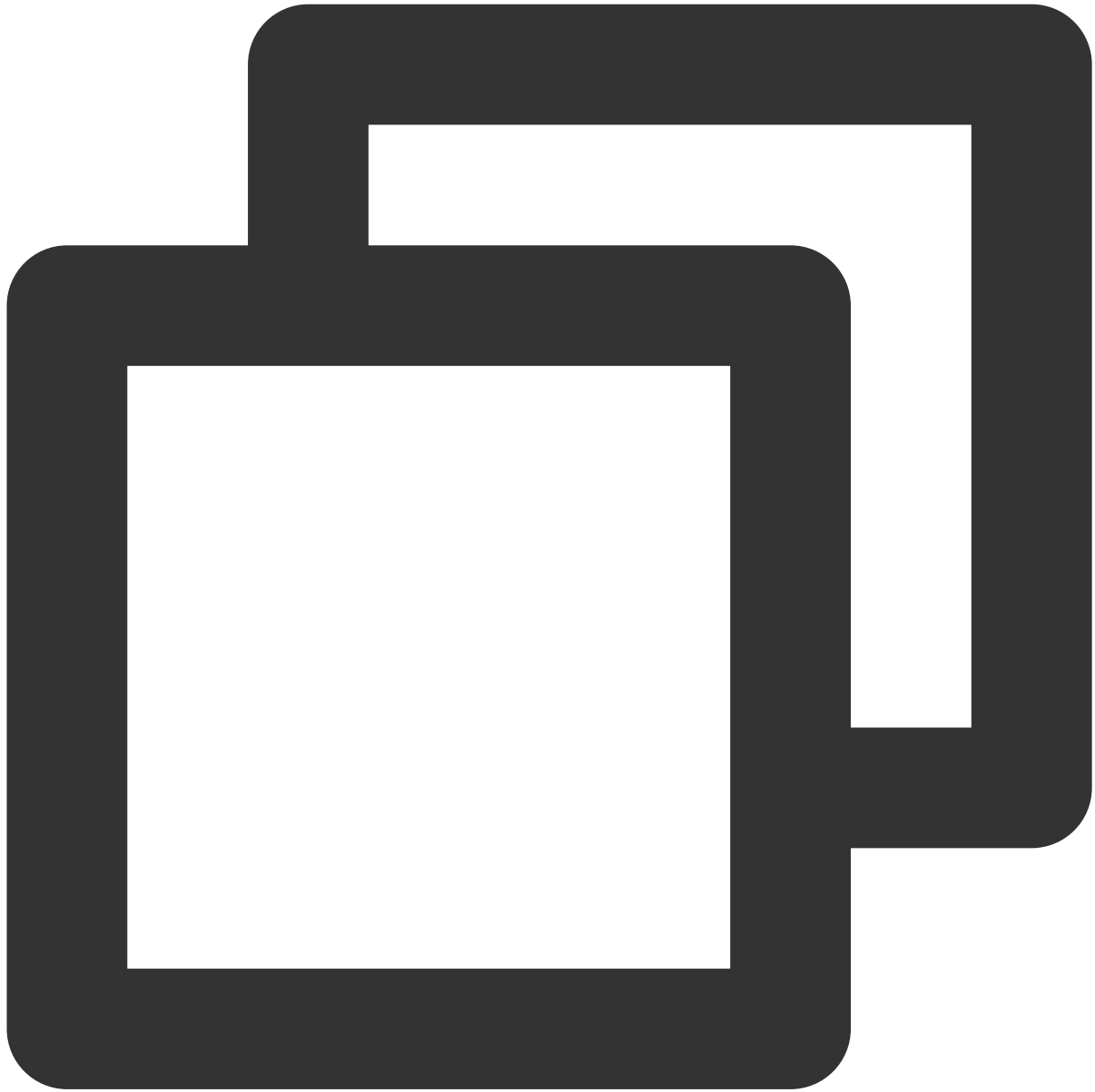
Listing versioned objects

COS stores object version information in the versions parameter associated with the bucket and returns object versions in the order of storage time from the newest to the oldest.

Querying all versions of a specific object

You can query all versions of a particular object using the versions parameter together with the prefix request parameter. For more information on prefix, see [GET Bucket Object Versions](#).

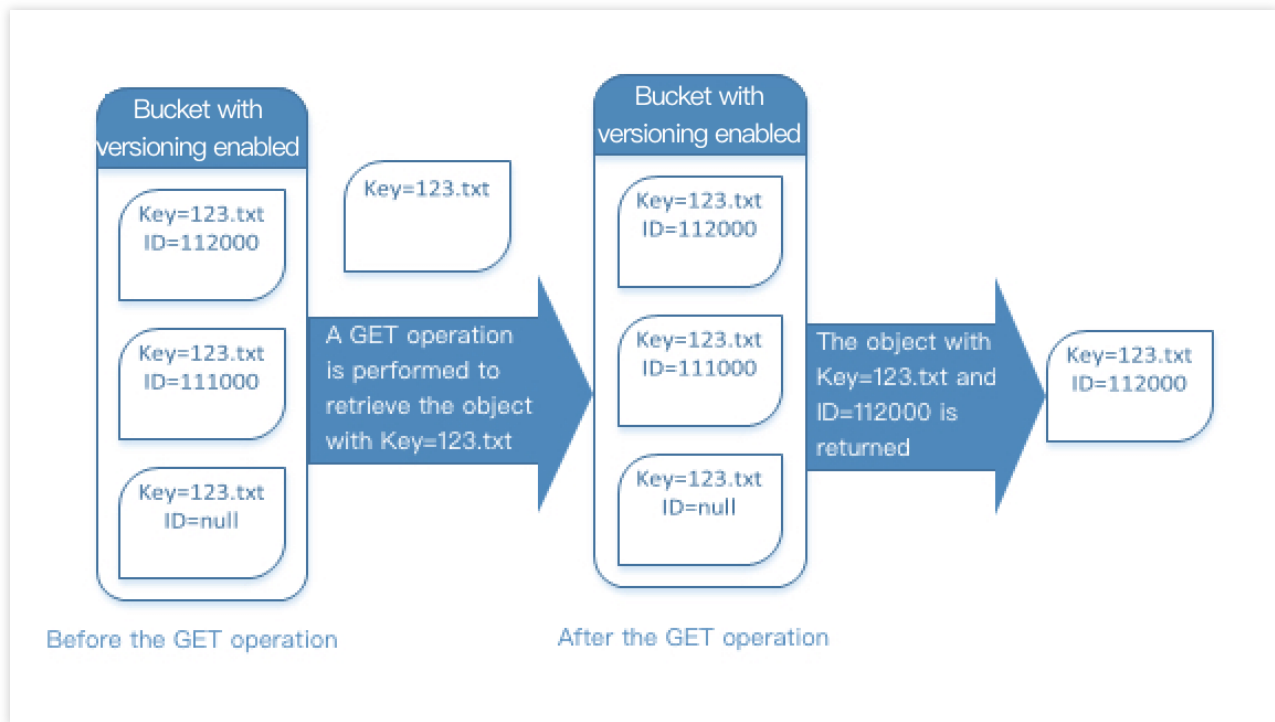
Below is a sample request for getting all versions of a specific object:



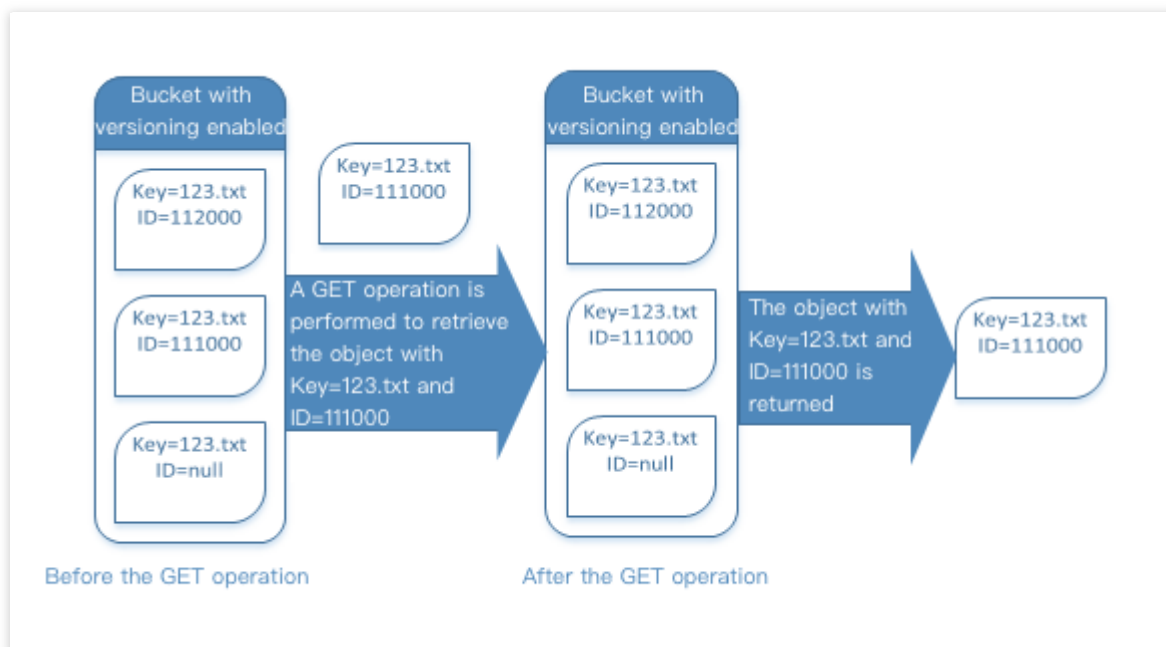
```
GET /?versions&prefix=ObjectKey HTTP/1.1
```

Querying an object version

A GET request with no version ID specified returns the current version of an object. The following figure shows how a GET request returns the current version (the latest version) of the `123.txt` object.



A GET request with a version ID specified returns the specified version of an object. The following figure shows how such a request returns the specified version of the object. The request can also return the current version.



Querying the metadata of an object version

If you only want to query the metadata of an object instead of its content, use the HEAD operation. By default, you get the metadata of the latest version. To query the metadata of a specific object version, specify its version ID when

submitting the request.

To query the metadata of an object version:

Set the version ID to the ID of the version of the object whose metadata you want to query.

Send a HEAD operation request with the version ID specified.

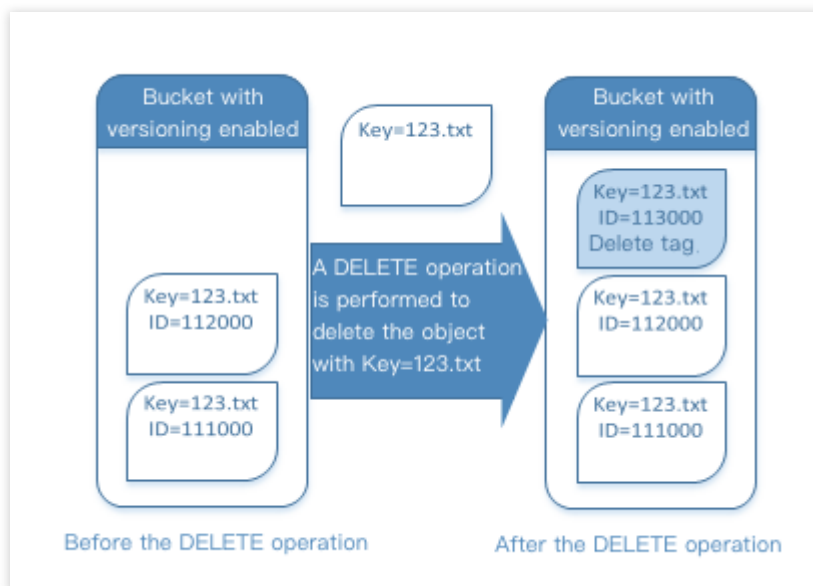
Deleting objects

You can delete unnecessary object versions. When you make a DELETE request in a versioning-enabled bucket, there will be two scenarios:

1. Perform a simple DELETE operation with no version ID specified.

This is similar to putting the deleted objects to the "recycle bin". The objects are not permanently deleted and can be restored if needed.

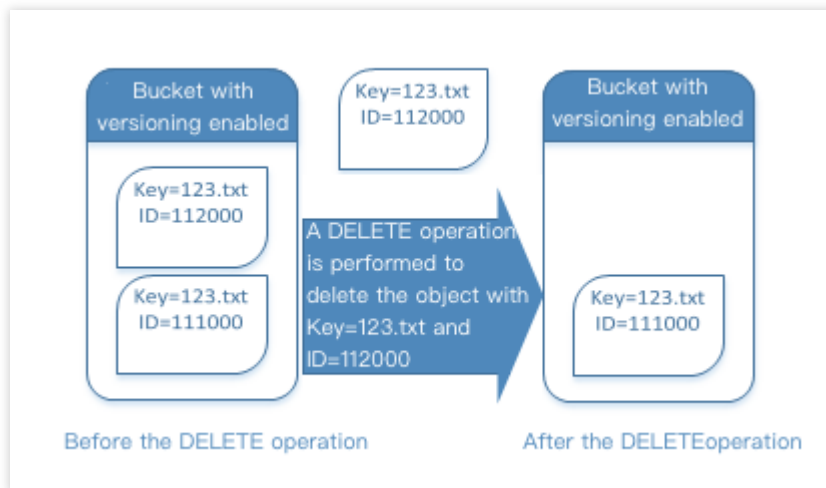
As you can see below, if you do not specify a version ID, the DELETE operation will not delete the 123.txt object; instead, it will insert a delete marker and add a new version ID.



Note:

COS will insert a delete marker with a new version ID for the deleted objects. The delete marker will become the current version of the deleted objects. When you try to GET the objects with a delete marker, COS will assume that the objects do not exist and will return a 404 error.

2. If you delete an object with a version ID specified, the specified version of the object will be deleted permanently.



Restoring Previous Versions

Versioning can be used to restore previous versions. You can do it in two ways:

1. Copy a previous version of the object into the same bucket

The copied object will become the current version of the object, and all object versions will be retained

2. Delete the current version of the object permanently

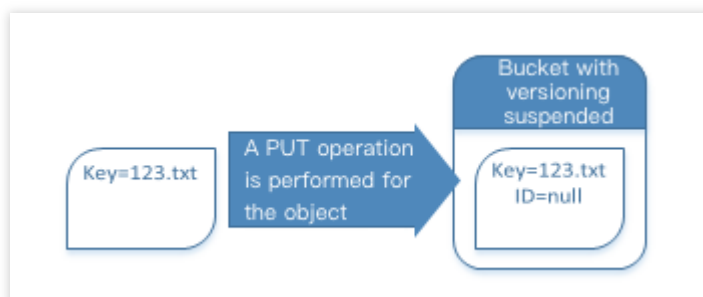
When you delete the current object version, you, in effect, turn the previous version into the current version of the object.

Managing objects in a versioning-suspended bucket

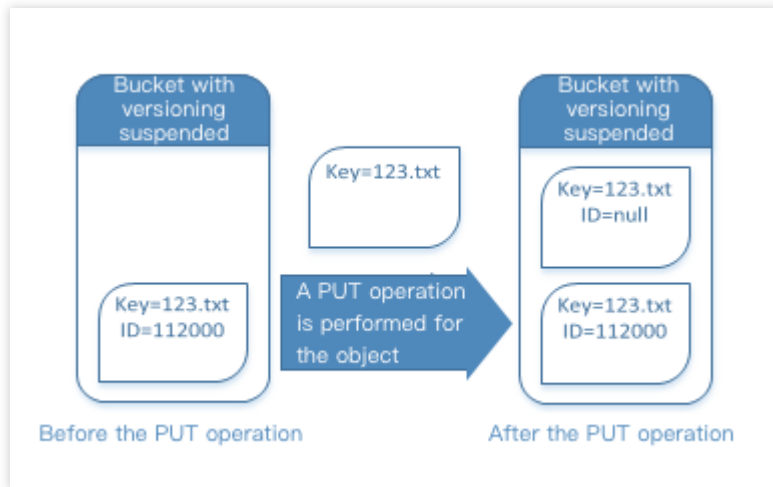
When you suspend versioning, existing objects in your bucket do not change. What changes is how COS handles objects in future requests. The section below describes how to manage objects in a bucket where versioning is suspended.

Uploading an object

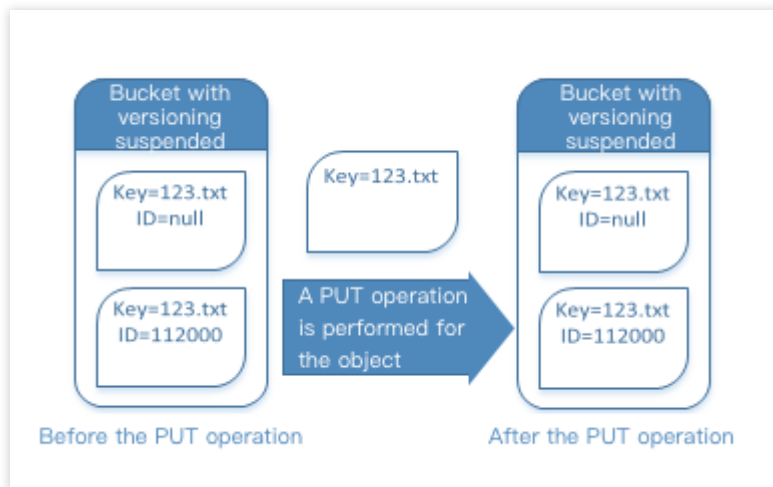
Once you suspend bucket versioning, when you use PUT, POST, or COPY operations, COS will automatically add a "null" version ID to objects added to the bucket thereafter as shown below:



If there are versioned objects in the bucket, the objects newly uploaded to the bucket will become the current version with a version ID of null as shown below:



If there is already a null version in the bucket, it will be overwritten, and the original object content will also be replaced accordingly as shown below:



Querying an object version

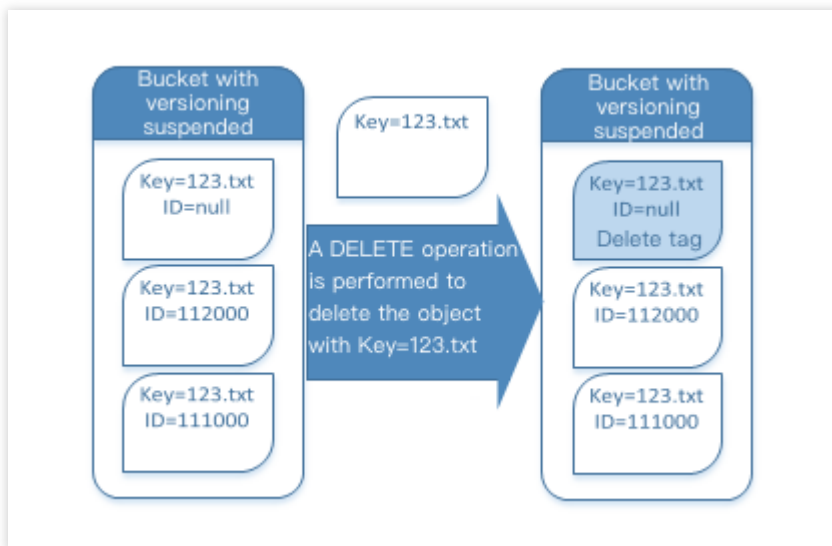
A GET Object request in a versioning-suspended bucket returns the current version of an object.

Deleting objects

If versioning is suspended, a DELETE request will result in one of the following:

If an object has a null version in the bucket, the null version of the object will be deleted.

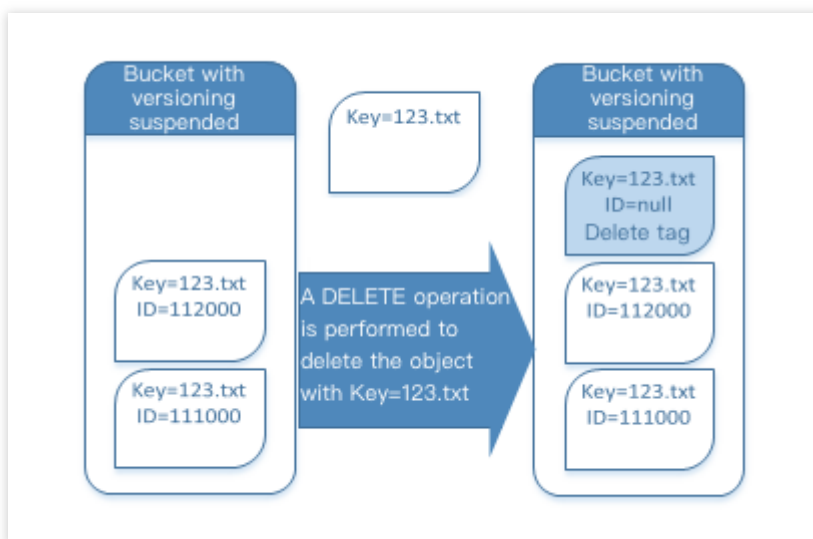
As you can see below, when you use a simple DELETE operation, COS will insert a delete marker for objects with a version ID of null.

**Note:**

Since a delete marker does not have content, you will lose the original content of the null version when a delete marker replaces it.

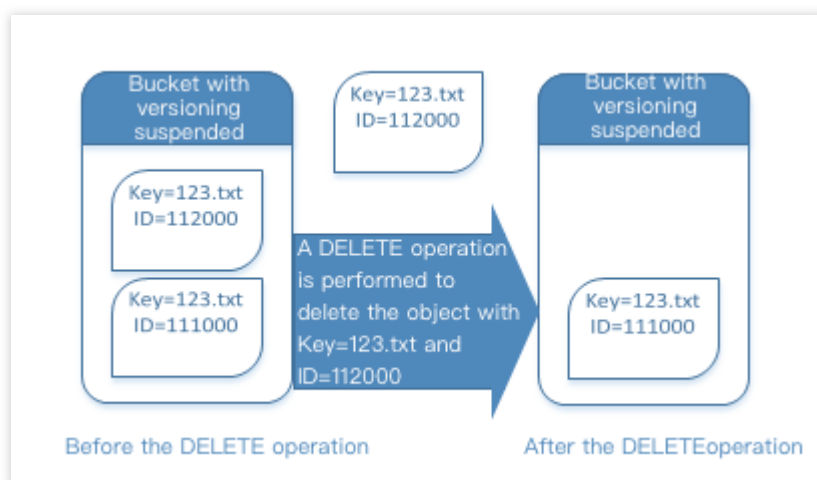
If an object does not have a null version in the bucket, a new delete marker will be added to the bucket.

As you can see below, if an object does not have a null version in the bucket, a simple DELETE will not remove anything; COS will just insert a delete marker.



Even in a bucket where versioning is suspended, the root account is still able to delete a specified version permanently.

The following figure shows that deleting a specified object version will delete the object permanently.

**Note:**

Only the root account or an account authorized by it can delete a specified object version.

Delete Markers

Last updated : 2024-03-25 15:33:39

Overview

A delete marker is used to mark a versioned object as “deleted” in COS. It has an object key and version ID just like an object, but with the following differences:

Its content is empty.

It is not associated with an ACL value.

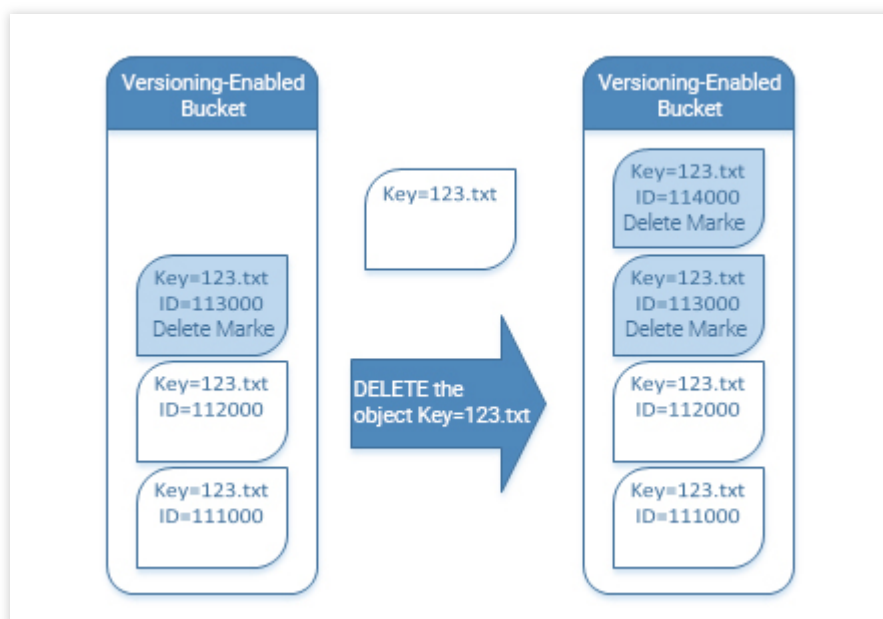
A GET request on a delete marker will return a 404 error.

The only operation you can use on a delete marker is DELETE, and only the root account can make such a request.

Deleting a Delete Marker

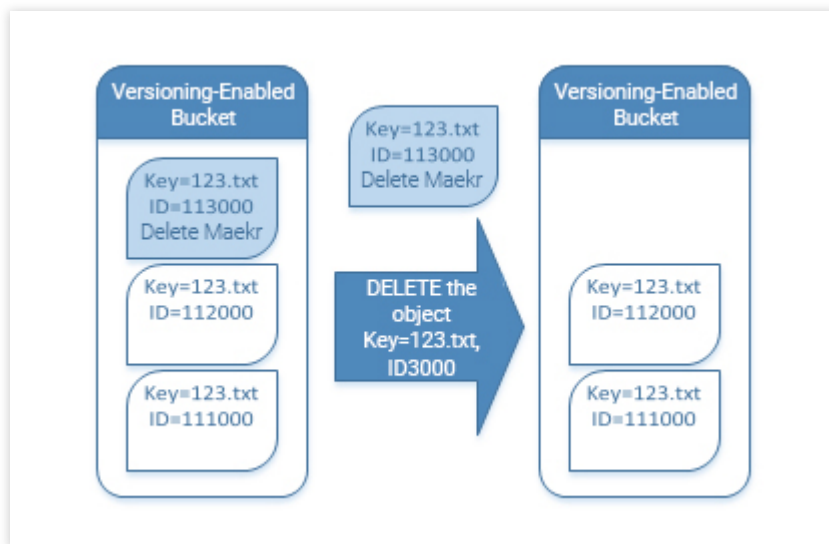
To permanently delete a delete marker, specify its version ID in a DELETE Object versionId request. If you use a DELETE request to delete a delete marker without specifying its version ID, COS will not delete the delete marker, but instead, insert another delete marker.

The following figure shows how a simple DELETE on a delete marker removes nothing, but adds a new delete marker to the bucket.



In a versioning-enabled bucket, a new delete marker has a unique version ID. Therefore, one object may have multiple delete markers in the same bucket. To delete a delete marker permanently, you must include its version ID in a DELETE Object versionId request.

The following figure shows how you can permanently delete a delete marker with a DELETE Object versionId request.



You can delete a delete marker only after the root account grants the `DeleteObject` permission.

To permanently delete a delete marker:

1. Set versionId to the ID of the version of the delete marker you want to delete.
2. Send a DELETE Object versionId request.

Using Versioning

Versioning Configuration

Last updated : 2024-03-25 15:33:39

Use Cases

Versioning allows you to store multiple versions of an object in a bucket and retrieve, delete, or restore a specified version.

For more information, see [Versioning Overview](#).

Note:

Only the root account and authorized sub-accounts can configure the versioning state of a bucket.

Directions

Using the COS console

For details, see [Setting Versioning](#).

Using the REST API

Use the REST API to configure versioning and manage objects in buckets with different versioning states. For more information, see the following API documentation:

[PUT Bucket versioning](#)

[GET Bucket versioning](#)

[GET Bucket Object versions](#)

[PUT Object](#)

[GET Object](#)

[DELETE Object](#)

[DELETE Multiple Objects](#)

Using the SDK

Call the Versioning method in the SDK. For more information, see the SDK documentation for the corresponding programming language below:

[Android SDK](#)

[C SDK](#)

[C++ SDK](#)

[.NET SDK](#)

[Go SDK](#)

[iOS SDK](#)

[Java SDK](#)

[JavaScript SDK](#)

[Node.js SDK](#)

[PHP SDK](#)

[Python SDK](#)

[Mini Program SDK](#)

Cross-Bucket Replication

Overview

Last updated : 2024-03-25 15:33:39

Overview

Cross-bucket replication enables COS to automatically and asynchronously replicate **incremental objects** from one bucket to another bucket. With cross-bucket replication, COS can accurately replicate the exact same objects, along with object metadata and version IDs, from a source bucket to a destination bucket. Additionally, object operations, such as adding or deleting an object, may also be synced to the destination bucket.

Note:

To enable cross-bucket replication, make sure that versioning is enabled for both the source and destination buckets. After cross-bucket replication is enabled, the object copies will be of the same storage class as the source objects, unless you specify a different storage class during replication.

During replication, COS will copy the access control list (ACL) of the source bucket. Currently, the source bucket and destination bucket must be owned by the same account.

Use Cases

Remote disaster recovery: COS offers 12 9's of durability for object data, but there is still a slight chance of data loss due to force majeure such as wars and natural disasters. To avoid data loss by explicitly having a separate copy in a different bucket, you can use cross-bucket replication that helps remote disaster recovery. In this way, when the IDC for one bucket is damaged due to force majeure, the IDC for the other bucket can still provide data copies for your use.

Compliance: COS ensures data availability by providing multiple copies and erasure codes for data in physical disks by default. However, some industries may have compliance requirements stipulating that you keep copies in another bucket. Cross-bucket replication allows data to be replicated across buckets to meet such requirements.

Minimizing access latency: when you have end users accessing objects from different regions, with cross-bucket replication, you can maintain object copies in the buckets closest to them. This minimizes access latency to deliver a better user experience.

Special technical requirements: if you have compute clusters in two different buckets and the clusters need to process the same set of data, with cross-bucket replication, you can maintain object copies in both buckets.

Data migration and backup: you can copy your business data from one bucket to another as needed for data migration and backup.

Notes

Replication time

The time it takes for COS to replicate objects between buckets may range from a few minutes to a few hours, depending on factors including:

Object size: it takes more time to replicate larger objects, for which multipart upload is recommended for faster upload and synchronization.

Distance between the source and destination buckets: A longer distance between the regions of the buckets requires more time to transfer data.

How the objects were uploaded: unlike simple upload where data can only be uploaded or downloaded serially, multipart upload supports concurrent uploads which can speed up upload and cross-bucket replication of large files.

For more information, please see [Simple Upload](#) and [Multipart Upload](#).

Lifecycle

Cross-bucket replication requires you to enable versioning first, which will keep multiple historical versions of objects in your buckets and result in more storage usage. COS cross-bucket replication will incur fees for requests, downstream traffic, and storage usage. Among them, storage usage is charged at the prices in the destination bucket's region. To reduce the costs or customize how to retain your data, use [lifecycle management](#) as needed.

If the copied objects in the destination bucket need to follow the same lifecycle rules as the source bucket, create the same rules for the destination bucket.

If lifecycle rules exist for the destination bucket, note that the creation time of the object copies generated by cross-bucket replication is the creation time of the source objects but not the time when the copies are replicated to the destination bucket.

If lifecycle rules exist for the source bucket and an object being replicated needs to be deleted according to the rules, the object will still be replicated, and the object copy will be retained in the destination bucket.

Versioning

To use cross-bucket replication, you need to enable versioning for both the source and destination buckets. For more information, please see [Versioning Overview](#). Once versioning is enabled, you should note that disabling it will affect cross-bucket replication:

If you try to disable versioning for a bucket where cross-bucket replication is enabled, COS will return an error prompting you to delete the cross-bucket replication rule before disabling versioning.

If you try to disable versioning for a destination bucket, COS will prompt you that doing so will affect cross-bucket replication. If you proceed to disable versioning, the cross-bucket replication rule that uses this bucket as the destination bucket will become invalid.

Actions

Last updated : 2024-03-25 15:33:39

This document describes what COS will and will not replicate after you enable cross-bucket replication for a bucket.

What is replicated

In a source bucket with cross-bucket replication enabled, COS will replicate the following:

Any new objects uploaded to the source bucket after the cross-bucket replication rule is added.

Object attributes such as object metadata and version ID.

Information on object operations such as adding an object of the same name (equivalent to adding a new object) and deleting an object.

Note:

If you specify an object version to delete in the source bucket by specifying a version ID, COS will not replicate this delete operation.

If you add a bucket-level configuration such as a lifecycle rule to the source bucket, COS will not replicate any resulting object operations.

Delete operations in cross-bucket replication

If an object is deleted from a source bucket with cross-bucket replication enabled, the following will happen:

If you try to delete an object from the source bucket without specifying a version ID, COS will add a delete marker to the object. Then, if you select the option **Sync Delete Marker**, cross-bucket replication will replicate this delete marker to the destination bucket. Regardless of whether the delete marker is replicated, the object will not be deleted from the destination bucket. You can always access a noncurrent version of the object by specifying its version ID. For more information, please see [Versioning Overview](#).

If you specify a version ID to delete an object, COS will delete the specified object version from the source bucket, but will not replicate the delete operation to the destination bucket. That is, COS will not delete this object version from the destination bucket. This prevents malicious data deletion.

What isn't replicated

In a source bucket with cross-bucket replication enabled, COS will not replicate the following:

Objects that existed before cross-bucket replication is enabled.

An object's encryption information, which will be lost once the encrypted object is replicated.

Object data in the source bucket that has been copied from another bucket.

Updates to bucket-level configuration.

Actions performed by lifecycle configuration.

Note:

The cross-bucket replication of object data between buckets is not transitive. If you set two cross-bucket replication rules, one of which sets bucket A as the source bucket and bucket B as the destination bucket, and the other sets bucket B as the source bucket and bucket C as the destination bucket, then the object data added to bucket A will be replicated only to bucket B but not to bucket C.

For example, if you update the lifecycle configuration of the source bucket, COS will not apply this configuration to the destination bucket synchronously.

If you configure a lifecycle rule only for the source bucket, COS will add delete markers to expired objects there, but the markers will not be replicated to the destination bucket. For the destination bucket to delete these expired objects as well, you need to configure the same lifecycle rule for the destination bucket.

Configuration

Last updated : 2024-03-25 15:33:39

Use Cases

Cross-bucket replication enables you to replicate objects from a source bucket to a destination bucket. This feature is well suited for use cases such as remote disaster recovery, industry compliance, data migration and backup, access latency reduction, and access to data located in different regions.

Special use cases:

Multi-region backup: You can configure multiple replication rules for the source bucket to replicate objects to buckets in different regions for multi-region backup and disaster recovery.

Two-way replication: You can create replication rules in the source and destination buckets respectively to implement two-way data replication.

Note:

Once versioning is enabled, you can create multiple versions for any new object uploaded. Each of these versions occupies your storage capacity and is billed for storage equally.

Directions

Using the COS console

You can configure a cross-bucket replication rule in the COS console as instructed in [Setting Cross-Bucket Replication](#).

Using RESTful APIs

You can configure and manage cross-bucket replication rules through RESTful APIs as described in the following API documents:

[PUT Bucket replication](#)

[GET Bucket replication](#)

[DELETE Bucket replication](#)

Using SDKs

You can directly call the cross-bucket replication method in SDKs. For more information, see the SDK documentation for the corresponding programming language below:

[Android SDK](#)

[C SDK](#)

[C++ SDK](#)

[.NET SDK](#)

[Go SDK](#)

[iOS SDK](#)

[Java SDK](#)

[JavaScript SDK](#)

[Node.js SDK](#)

[PHP SDK](#)

[Python SDK](#)

MAZ Feature Overview

Last updated : 2024-03-25 15:33:39

MAZ refers to the multi-AZ storage architecture offered by [COS](#), which can provide IDC-level disaster recovery capabilities for your data.

Your data is scattered among multiple IDCs in a region. When an IDC fails due to extreme situations such as natural disasters or power failures, the multi-AZ storage architecture can still provide stable and reliable storage services. Multi-AZ provides 99.999999999% (12 nines) design data reliability and 99.995% design service availability. When you upload data objects to COS, you can store them in a multi-AZ region simply by specifying the storage class.

Note:

Currently, the MAZ configuration of COS is supported only in Beijing, Guangzhou, Shanghai, Hong Kong (China), Singapore regions and will be available in other public cloud regions in the future.

Using the MAZ configuration incurs high storage usage fees. For more information, see [Pricing | Cloud Object Storage](#).

Strengths of Multi-AZ

If you store your data in a multi-AZ region, the data will be divided into multiple chunks, and corresponding coding chunks will be calculated based on the erasure code algorithm. The original data chunks and coding chunks will be mixed up and evenly distributed to different IDCs in the region for storage and intra-region disaster recovery. When an IDC becomes unavailable, the data can still be read from or written to other IDCs normally, ensuring that your data can be persistently stored without loss, which maintains your business data continuity and high availability. The COS multi-AZ feature has the following strengths:

Intra-region disaster recovery: cross-IDC disaster recovery is supported. In the multi-AZ storage architecture, object data is stored on different devices in different IDCs in the same region. When an IDC fails, other redundant IDCs remain available, guaranteeing that your business will not be affected, and data will not be lost.

Stability and durability: the erasure code-based redundant storage mechanism is leveraged to provide up to 99.999999999% design data reliability. Data is stored in chunks and read and written concurrently to provide up to 99.995% design service availability.

Ease of use: you can specify the storage architecture for your data by specifying the object storage class. You can even select any objects in a bucket and store them in the multi-AZ architecture for greater ease of use.

Multi-AZ storage and non-multi-AZ storage are compared below for their specifications and limitations:

Comparison Item	MAZ Storage	Non-MAZ Storage
Designed data	99.999999999% (12 nines)	99.999999999% (11

durability		
Designed service availability	99.995%	99.99%
Supported regions	See Storage Class Overview .	
Supported storage classes	MAZ_STANDARDMAZ_STANDARD_IAMAZ_INTELLIGENT_TIERING	STANDARDSTANDARD_ARCHIVEINTELLIGENT_TIERING

How to Use

You can enable MAZ for a bucket and set the object storage class to MAZ for objects uploaded to it. When uploading an object, you can store it in the MAZ storage architecture by simply specifying the object storage class.

In short, you only need to perform the following two steps to store files in the multi-AZ architecture:

1. Create a bucket as instructed in [Creating Bucket](#) and enable the MAZ configuration **during creation**.
2. Upload a file and specify the storage class during the upload. For more information about how to upload a file, see [Uploading Objects](#).

Note:

Once multi-AZ is enabled for the bucket, it cannot be disabled, so please enable it with caution. Multi-AZ is not enabled for existing buckets by default, and it can be enabled only for new buckets.

For a bucket with MAZ configuration enabled, objects can be uploaded to MAZ storage classes (MAZ_STANDARD, MAZ_STANDARD_IA, or MAZ_INTELLIGENT_TIERING). To upload an object to MAZ_INTELLIGENT_TIERING, the intelligent tiering configuration must also be enabled for the bucket.

If you want to store existing data in an MAZ bucket, you can create a bucket with MAZ enabled and use the batch replication feature of COS Batch to replicate files in the existing bucket to the new bucket in batches. For more information on how to use COS Batch, see [Batch Operation](#).

Use Limits

Currently, COS allows you to upload objects to the MAZ_STANDARD, MAZ_STANDARD_IA, or MAZ_INTELLIGENT_TIERING storage class. Therefore, there are also restrictions on features related to storage class changes as detailed below:

Storage class limit: Currently, objects can be uploaded only to MAZ storage classes (MAZ_STANDARD, MAZ_STANDARD_IA, or MAZ_INTELLIGENT_TIERING). To upload an object to MAZ_INTELLIGENT_TIERING, both the MAZ configuration and intelligent tiering configuration must be enabled for the bucket.

Operation limit: currently, objects can only be uploaded, downloaded, and deleted. Objects can be replicated to a multi-AZ bucket but not to a single-AZ bucket.

Lifecycle limit: Currently, objects can be deleted only upon expiration but cannot be transitioned from an MAZ storage class to an OAZ storage class.

Cross-region replication limit: Objects cannot be replicated from an MAZ storage class to an OAZ storage class.

Data Security

Server-side Encryption Overview

Last updated : 2024-03-25 15:33:39

Overview

Cloud Object Storage (COS) encrypts your data at the object level before data is written to disks and automatically decrypts the data when you access it. Encryption and decryption are completed on servers. Server-side encryption can effectively protect static data.

Note:

As long as you have access permission on an object, your object accessing experience is the same regardless of whether the object is encrypted.

Server-side encryption encrypts only the object data but not its metadata. Server-side encrypted objects can only be accessed with a valid signature and cannot be accessed by anonymous users.

Use Cases

Private data storage: For private data storage, server-side encryption can encrypt stored data to protect your privacy and automatically decrypt the data when you access it.

Private data transfer: For private data transfer, COS supports deploying SSL certificates with HTTPS to implement encryption. An encryption layer will be established on the transfer linkage layer, ensuring that data will not be stolen or tampered with during transfer.

Encryption

COS supports multiple server-side encryption methods such as SSE-COS and SSE-C. You can choose the appropriate one to encrypt data stored in COS.

SSE-COS Encryption

SSE-COS: Server-side encryption with a key managed by COS. In this mode, COS will manage the master key and data, and users can manage and encrypt the data directly through COS. SSE-COS uses a strong AES-256 multi-factor encryption to ensure that each object is encrypted with a unique key, while regularly rotating the master key to encrypt the key itself.

Note:

When uploading an object using the `POST` operation, you need to provide the same information in the form field instead of the `x-cos-server-side-encryption` header. For more information, see [POST Object](#).
SSE-COS encryption is not available for objects uploaded with a pre-signed URL. Instead, you need to use the COS console or HTTP request header to specify server-side encryption.

Using COS console

For details on how to use the COS console to perform SSE-COS encryption on objects, see [Setting Object Encryption](#).

Using RESTful APIs

Note:

When you list objects in a bucket, all objects will be listed, regardless of whether they are encrypted.

When uploading an object using the `POST` operation, provide the same information in the form field instead of the request header. For more information, see [POST Object](#).

When you request the following APIs, you can apply server-side encryption by providing the `x-cos-server-side-encryption` header. For more information, see [Common Request Headers - SSE-COS](#).

[PUT Object](#)

[Initiate Multipart Upload](#)

[PUT Object - Copy](#)

[POST Object](#)

SSE-KMS Encryption

SSE-KMS encryption is server-side encryption using a key managed by KMS. KMS is a security management service launched by Tencent Cloud, using a third-party-certified hardware security module (HSM) to generate and protect keys. KMS allows users to easily create and manage keys, meeting their key management needs for multiple applications and services, while satisfying regulatory and compliance requirements.

When using SSE-KMS encryption for the first time, you need to [enable the KMS service](#). After the KMS service is enabled, the system will automatically create a default customer master key (CMK) for you. You can also create your own keys through the [KMS console](#), and define key policies and use methods. KMS allows users to choose their own key material from **KMS** or **external** sources. For more information, see [Create Key](#) and [Import External Key](#).

Note:

SSE-KMS only encrypts the object data, not its metadata.

Currently, SSE-KMS only supports Beijing, Shanghai, Guangzhou, and Hong Kong (China) regions.

Using SSE-KMS encryption will incur an additional cost, which will be charged by KMS. For more information, see [KMS Billing Overview](#).

Objects encrypted with SSE-KMS can only be accessed with a valid signature but not by anonymous users.

Using COS console

For details on how to use the COS console to perform SSE-KMS encryption on objects, see [Setting Object Encryption](#).

Using RESTful APIs

Note:

When you list objects in a bucket, all objects will be listed, regardless of whether they are encrypted.

When uploading an object using the POST operation, please provide the same information in the form field instead of the request header. For more information, see [POST Object](#).

When you request the following APIs, you can apply server-side encryption by providing the `x-cos-server-side-encryption` header. For more information, see [Common Request Headers - SSE-KMS](#).

[PUT Object](#)

[Initiate Multipart Upload](#)

[PUT Object - Copy](#)

[POST Object](#)

Details

If you have never used **COS console** for SSE-KMS encryption, and only used **API** for SSE-KMS encryption, you need to create a [CAM role](#) first:

1. Log in to the CAM Console and go to the [Roles](#) page.
2. Click **Create Role** and select **Tencent Cloud Product Service** as the role entity.
3. Select **Cloud Object Storage** as the service supporting the role, and click **Next**.
4. Search for the **QcloudKMSAccessForCOSRole** role policy and select it, and click **Next**.

Enter role entity info

Configure role policy

3 Review

Policy List (1 in total)

QcloudKMSCreatorFullAccess

Policy Name	Policy Type
<input checked="" type="checkbox"/> <div>QcloudKMSCreatorFullAccess Key Management Service (KMS) resources creator's full read-write access to its resources</div>	Preset policy

(1) selected

Policy Name	Policy Type
QcloudKMSCreatorFullAccess Key Management Service (KMS) resources creator's full read-write access to its resources	Preset policy

Press Shift to select multiple items

Back

Next

5. Set the role tag keys and values and click **Next**.

6. Enter the specified role name: COS_QcsRole.

7. Click **Done** to complete the process.

SSE-C Encryption

SSE-C encryption is server-side encryption with a user-defined key. When you upload an object, COS will use the encryption key you provide to apply AES-256 encryption to your data.

Note:

COS does not store your encryption key. Instead, it stores the HMAC value of the encryption key with random data added, which is used to verify your request to access the object. COS cannot use the HMAC value to derive the value of the encryption key or decrypt the encrypted object. Therefore, if you lose the encryption key, the object cannot be obtained again.

When uploading an object using the POST operation, you need to provide the same information in the form field instead of the `x-cos-server-side-encryption-*` header. For more information, see [POST Object](#).

SSE-C can only be used via APIs but not the console.

Using RESTful APIs

When you request the following APIs, you can apply server-side encryption to PUT and POST requests by providing the `x-cos-server-side-encryption-*` header. For GET and HEAD requests, you need to provide the `x-`

`cos-server-side-encryption-*` header to decrypt the specified object encrypted with SSE-C. For more information, see [Common Request Headers - SSE-C](#). The following operations support this header:

[GET Object](#)

[HEAD Object](#)

[PUT Object](#)

[Initiate Multipart Upload](#)

[Upload Part](#)

[POST Object](#)

[PUT Object - Copy](#)

Bucket Encryption Overview

Last updated : 2024-03-25 15:33:39

Overview

By setting bucket encryption, you can encrypt all new objects uploaded to a bucket with the specified encryption method by default.

Currently, SSE-COS encryption is supported, i.e., server-side encryption that uses COS to manage keys.

For more information on server-side encryption, see [Server-Side Encryption Overview](#).

Directions

Using the COS console

You can set bucket encryption in the COS console as instructed in [Setting Bucket Encryption](#).

Using RESTful APIs

You can configure bucket encryption by using the following APIs:

[PUT Bucket encryption](#)

[GET Bucket encryption](#)

[DELETE Bucket encryption](#)

Notes

Uploading object to encrypted bucket

For buckets requiring encryption, note the following:

Configuring encryption for a bucket will not lead to encryption operations on objects that already exist in it.

After encryption is configured for a bucket, for objects uploaded to the bucket:

If your PUT request does not contain encryption information, the uploaded objects will be encrypted based on the encryption configuration of the bucket.

If your PUT request contains encryption information, the uploaded objects will be encrypted based on the contained encryption information.

After encryption is configured for a bucket, for inventory reports delivered to the bucket:

If encryption is not configured for the inventory, the delivered reports will be encrypted based on the encryption configuration of the bucket.

If encryption is configured for the inventory, the delivered reports will be encrypted based on the encryption configuration of the inventory.

After encryption is configured for a bucket, the data retrieved from the origin to the bucket will be encrypted based on the encryption configuration of the bucket by default.

Encrypting a bucket that has a cross-region replication rule configured

For the destination bucket that has a cross-region replication rule configured, if you configure encryption for it, note the following:

If the objects in the source bucket are not encrypted, the object copies in the destination bucket will be encrypted by default.

If the objects in the source bucket are encrypted, the object copies in the destination bucket will inherit the encryption from the source bucket, and the bucket encryption settings will not be applied.

Server-side Encryption Overview

Last updated : 2024-03-25 15:33:39

Overview

Cloud Object Storage (COS) encrypts your data at the object level before data is written to disks and automatically decrypts the data when you access it. Encryption and decryption are completed on servers. Server-side encryption can effectively protect static data.

Caution

As long as you have access permission on an object, your object accessing experience is the same regardless of whether the object is encrypted.

Server-side encryption encrypts only the object data but not its metadata. Server-side encrypted objects can only be accessed with a valid signature and cannot be accessed by anonymous users.

Use Cases

Private data storage: For private data storage, server-side encryption can encrypt stored data to protect your privacy and automatically decrypt the data when you access it.

Private data transfer: For private data transfer, COS supports deploying SSL certificates with HTTPS to implement encryption. An encryption layer will be established on the transfer linkage layer, ensuring that data will not be stolen or tampered with during transfer.

Encryption

COS supports multiple server-side encryption methods such as SSE-COS and SSE-C. You can choose the appropriate one to encrypt data stored in COS.

SSE-COS Encryption

SSE-COS: Server-side encryption with a key managed by COS. In this mode, COS will manage the master key and data, and users can manage and encrypt the data directly through COS. SSE-COS uses a strong AES-256 multi-factor encryption to ensure that each object is encrypted with a unique key, while regularly rotating the master key to encrypt the key itself.

Caution

When uploading an object using the `POST` operation, you need to provide the same information in the form field instead of the `x-cos-server-side-encryption` header. For more information, see [POST Object](#).
SSE-COS encryption is not available for objects uploaded with a pre-signed URL. Instead, you need to use the COS console or HTTP request header to specify server-side encryption.

Using COS console

For details on how to use the COS console to perform SSE-COS encryption on objects, see [Setting Object Encryption](#).

Using RESTful APIs

Caution

When you list objects in a bucket, all objects will be listed, regardless of whether they are encrypted.

When uploading an object using the `POST` operation, provide the same information in the form field instead of the request header. For more information, see [POST Object](#).

When you request the following APIs, you can apply server-side encryption by providing the `x-cos-server-side-encryption` header. For more information, see [Common Request Headers - SSE-COS](#).

[PUT Object](#)

[Initiate Multipart Upload](#)

[PUT Object - Copy](#)

[POST Object](#)

SSE-KMS Encryption

SSE-KMS encryption is server-side encryption using a key managed by KMS. KMS is a security management service launched by Tencent Cloud, using a third-party-certified hardware security module (HSM) to generate and protect keys. KMS allows users to easily create and manage keys, meeting their key management needs for multiple applications and services, while satisfying regulatory and compliance requirements.

When using SSE-KMS encryption for the first time, you need to [enable the KMS service](#). After the KMS service is enabled, the system will automatically create a default customer master key (CMK) for you. You can also create your own keys through the [KMS console](#), and define key policies and use methods. KMS allows users to choose their own key material from **KMS** or **external** sources. For more information, see [Create Key](#) and [Import External Key](#).

Caution

SSE-KMS only encrypts the object data, not its metadata.

Currently, SSE-KMS only supports Beijing, Shanghai, Guangzhou, and Hong Kong (China) regions.

Using SSE-KMS encryption will incur an additional cost, which will be charged by KMS. For more information, see [KMS Billing Overview](#).

Objects encrypted with SSE-KMS can only be accessed with a valid signature but not by anonymous users.

Using COS console

For details on how to use the COS console to perform SSE-KMS encryption on objects, see [Setting Object Encryption](#).

Using RESTful APIs

Caution

When you list objects in a bucket, all objects will be listed, regardless of whether they are encrypted.

When uploading an object using the POST operation, please provide the same information in the form field instead of the request header. For more information, see [POST Object](#).

When you request the following APIs, you can apply server-side encryption by providing the `x-cos-server-side-encryption` header. For more information, see [Common Request Headers - SSE-KMS](#).

[PUT Object](#)

[Initiate Multipart Upload](#)

[PUT Object - Copy](#)

[POST Object](#)

Details

If you have never used **COS console** for SSE-KMS encryption, and only used **API** for SSE-KMS encryption, you need to create a [CAM role](#) first:

1. Log in to the CAM Console and go to the [Roles](#) page.
2. Click **Create Role** and select **Tencent Cloud Product Service** as the role entity.
3. Select **Cloud Object Storage** as the service supporting the role, and click **Next**.
4. Search for the **QcloudKMSAccessForCOSRole** role policy and select it, and click **Next**.

Enter role entity info

Configure role policy

3 Review

Policy List (1 in total)

QcloudKMSCreatorFullAccess

Policy Name	Policy Type
<input checked="" type="checkbox"/> <div>QcloudKMSCreatorFullAccess Key Management Service (KMS) resources creator's full read-write access to its resources</div>	Preset policy

(1) selected

Policy Name	Policy Type
QcloudKMSCreatorFullAccess Key Management Service (KMS) resources creator's full read-write access to its resources	Preset policy

Press Shift to select multiple items

Back

Next

5. Set the role tag keys and values and click **Next**.

6. Enter the specified role name: COS_QcsRole.

7. Click **Done** to complete the process.

SSE-C Encryption

SSE-C encryption is server-side encryption with a user-defined key. When you upload an object, COS will use the encryption key you provide to apply AES-256 encryption to your data.

Caution

COS does not store your encryption key. Instead, it stores the HMAC value of the encryption key with random data added, which is used to verify your request to access the object. COS cannot use the HMAC value to derive the value of the encryption key or decrypt the encrypted object. Therefore, if you lose the encryption key, the object cannot be obtained again.

When uploading an object using the POST operation, you need to provide the same information in the form field instead of the `x-cos-server-side-encryption-*` header. For more information, see [POST Object](#).

SSE-C can only be used via APIs but not the console.

Using RESTful APIs

When you request the following APIs, you can apply server-side encryption to PUT and POST requests by providing the `x-cos-server-side-encryption-*` header. For GET and HEAD requests, you need to provide the `x-`

`cos-server-side-encryption-*` header to decrypt the specified object encrypted with SSE-C. For more information, see [Common Request Headers - SSE-C](#). The following operations support this header:

[GET Object](#)

[HEAD Object](#)

[PUT Object](#)

[Initiate Multipart Upload](#)

[Upload Part](#)

[POST Object](#)

[PUT Object - Copy](#)

Object Lock Overview

Last updated : 2024-03-25 15:33:39

Overview

Tencent Cloud Object Storage (COS) offers the object lock function, enabling users to set a retention period during which objects cannot be modified or deleted.

The object lock function is currently implemented at the object level, allowing users to set a retention period for one or multiple objects. Users can also enable the object lock configuration for a bucket, thereby setting a default object lock for objects newly uploaded to the bucket.

Note:

Once an object is set with object lock, it does not support the removal of the object lock, nor does it support the reduction of the retention period. It only permits the extension of the retention period.

Note:

Based on the object lock function, COS can meet stringent electronic record retention requirements, including compliance with SEC Rule 17a-4(f), FINRA 4511, CFTC 1.31, and other regulatory standards.

SEC 17A-4 is a regulation promulgated by the U.S. Securities and Exchange Commission pursuant to the U.S. Securities Exchange Act of 1934. This regulation outlines data retention, indexing, and accessibility requirements applicable to companies engaged in financial securities trading or brokerage businesses, such as stocks, bonds, and futures. According to this regulation, many types of transaction records must be retained and cannot be overwritten or erased, with a period of immediate access within two years and non-immediate access for at least six years.

Enabling the Object Lock Function for a Bucket

Before locking an object, you need to enable the write once read many (WORM) function in the bucket. Note that enabling the object lock function merely marks the bucket as the status of "capable of using the object lock function", it does not automatically lock all objects in the bucket. Once the WORM function is enabled for the bucket, users can choose to enable the object lock function for specific objects, or maintain an unlocked state.

Once the object lock configuration for a bucket is enabled, it cannot be disabled. Buckets with the object lock configuration enabled will be subject to the following restrictions:

Enabling version control is not supported. If version control is already enabled for the bucket, the object lock configuration cannot be enabled.

Enabling bucket replication is not supported. This is because bucket replication requires version control to be enabled.

Enabling intelligent tiering configuration is not supported. If intelligent tiering configuration is already enabled for the bucket, the object lock configuration cannot be enabled.

The use of append upload is not supported.

How to Use

You can enable the object lock configuration for your bucket via APIs.

Using REST APIs

Invoke the [PUT Bucket ObjectLockConfiguration](#) interface to enable the object lock configuration.

Setting a Retention Period for an Object

Retention Period

Once the object lock function is enabled for a bucket, users can configure a lock retention period for the objects in the bucket. COS stores a timestamp in the object's metadata to denote the expiration of the lock (RetainUntilDate). Before the retention period:

The object cannot be deleted or modified.

The storage type of the object cannot be modified.

The HTTP headers and user metadata of the object cannot be modified, including **Content-Type**, **Content-Encoding**, **Content-Language**, **Content-Disposition**, **Cache-Control**, **Expires**, and **x-cos-meta-**.

Users can set the retention period for a specified object. During the object uploading process, the retention period can be configured through the `x-cos-object-lock-retain-until-date` header; after the object is uploaded, the retention period can be set via the `PUT Object Retention` interface.

Users can also set the default configuration for the bucket to automatically configure the retention period during the object uploading process. For more details, refer to [the default configuration of the bucket](#).

How to Use

You can set a retention period for an object through APIs.

Using REST APIs

Set a retention period for an object during the uploading process.

Invoke the uploading interfaces (PutObject, InitiateMultipartUploads, and PostObject), and simultaneously pass in the object lock-related headers or form fields. For more details, refer to the following API documentation:

[PUT Object](#)

[POST Object](#)

[Initiate Multipart Upload](#)

Set a retention period for the object after it is uploaded. For more details, refer to [PUT Object Retention](#).

View the retention period of the object. For more details, refer to [GET Object Retention](#).

Setting a Default Retention Period for a Bucket

COS supports adding a default object lock retention period for a bucket. The period is measured in days. When a user uploads an object without carrying an object lock header, the object will automatically be locked according to the retention period configured in the bucket. However, when a user uploads an object carrying an object lock header, the bucket configuration will be ignored, and the retention period will be set according to the header.

Note:

The default retention period for a bucket can be shortened, extended, or disabled.

Configuring the default retention period for a bucket will only affect subsequently added objects and will not impact the lock status of existing objects.

The following examples illustrate the impact of the default bucket configuration on the retention period of objects.

Object Upload Time	Default Bucket Configuration	Header During the Object Uploading Process	Object Lock Expiration Time
2023-01-01T18:30:00Z	7 days	x-cos-object-lock-retain-until-date: 2023-01-02T18:30:00Z	2023-01-02T18:30:00Z
2023-01-01T18:30:00Z	7 days	x-cos-object-lock-retain-until-date: 2023-01-20T18:30:00Z	2023-01-20T18:30:00Z
2023-01-01T18:30:00Z	7 days	Without carrying	2023-01-08T18:30:00Z
2023-01-01T18:30:00Z	Unconfigured	x-cos-object-lock-retain-until-date: 2023-01-02T18:30:00Z	2023-01-02T18:30:00Z
2023-01-01T18:30:00Z	Unconfigured	x-cos-object-lock-retain-until-date: 2023-01-20T18:30:00Z	2023-01-20T18:30:00Z
2023-01-01T18:30:00Z	Unconfigured	Without carrying	Unlocked

How to Use

You can set the default configuration of object lock for a bucket through APIs.

REST APIs

You can manage the object lock directly through the following APIs:

[PUT Bucket ObjectLockConfiguration](#)

[GET Bucket ObjectLockConfiguration](#)

Limitations

1. The object lock function is currently only available to customers in the allowlist. For details, [contact us](#).
2. The object lock function does not support the version control function. Buckets with the object lock function enabled do not support the enabling of version control, and buckets with version control in an enabled or paused state do not support the enabling of the object lock function.
3. The object lock function does not support the bucket replication function. As the object lock function is not compatible with the version control function, and the source and target buckets must have version control enabled under the bucket replication rules, buckets with the object lock function enabled cannot be used as the source or target buckets for bucket replication.
4. The object lock function does not support the intelligent tiering function. Buckets with the object lock function enabled do not support the enabling of intelligent tiering configuration, and once the intelligent tiering configuration is enabled in a bucket, it does not support the enabling of the object lock function.
5. The relationship between the object lock function and lifecycle rules is as follows. When an object is within its retention period, the sedimentation and deletion operations configured in the lifecycle do not take effect; once the retention period of an object expires, the sedimentation and deletion operations configured in the lifecycle can function normally.

Lifecycle Rule	Object Lock Retention Period	Lifecycle Execution Performance
Object sedimentation after 20 days	30-day retention period	On the 20th day after the upload, the object does not undergo sedimentation. On the 31st day after the upload, the object undergoes sedimentation.
Object deletion after 20 days	30-day retention period	On the 20th day after the upload, the object is not deleted. It is deleted on the 31st day after the upload.
Object sedimentation after 20 days	10-day retention period	The object undergoes sedimentation normally 20 days after the upload.
Object deletion after 20 days	10-day retention period	The object is deleted normally 20 days after the upload.

6. After the object lock function is enabled for a bucket, file fragments are not subject to object lock rules, allowing users to clear file fragments in the bucket.

7. Once the object lock function is enabled, it cannot be disabled.
8. After the object lock function is enabled, the bucket and object ACL can be modified.

Cloud Access Management

Access Permission Configuration Description

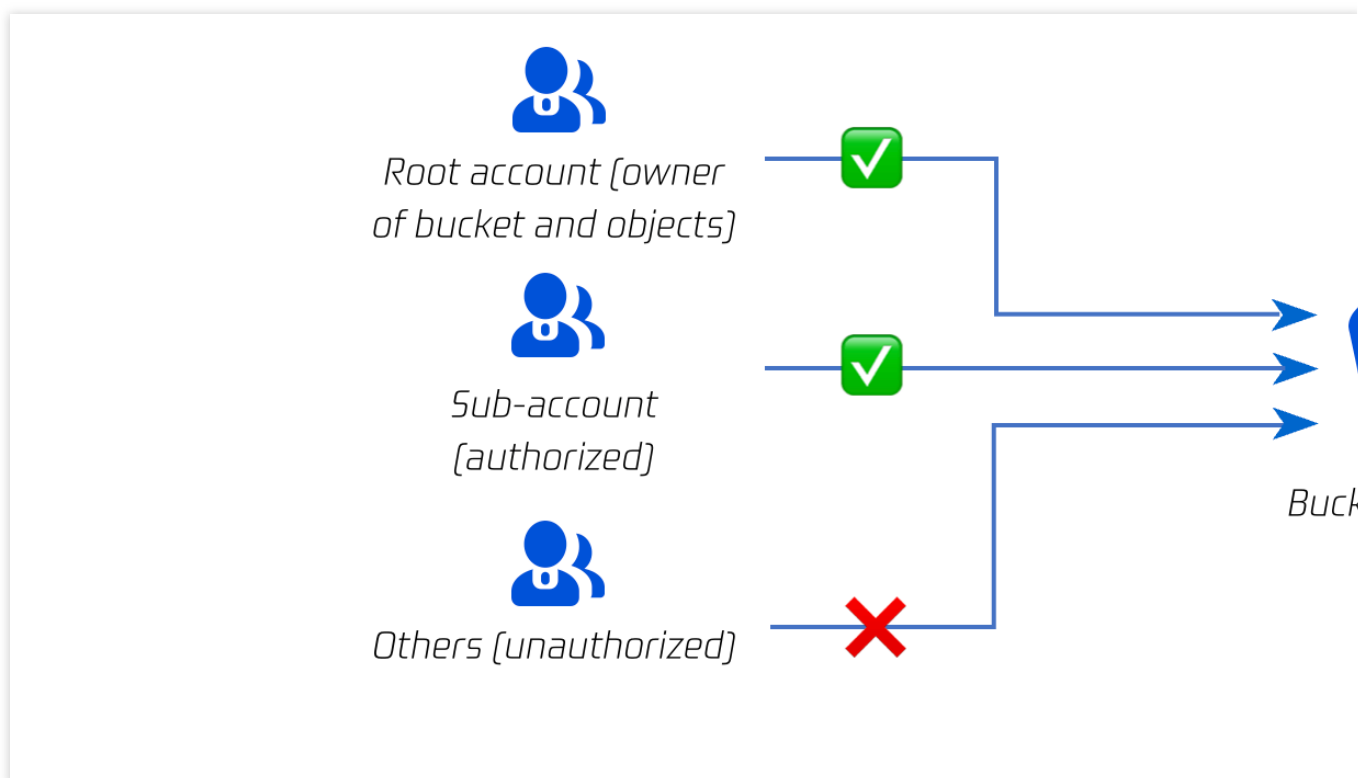
Access Control Overview

Basic Concepts of Access Control

Last updated : 2024-03-25 15:33:39

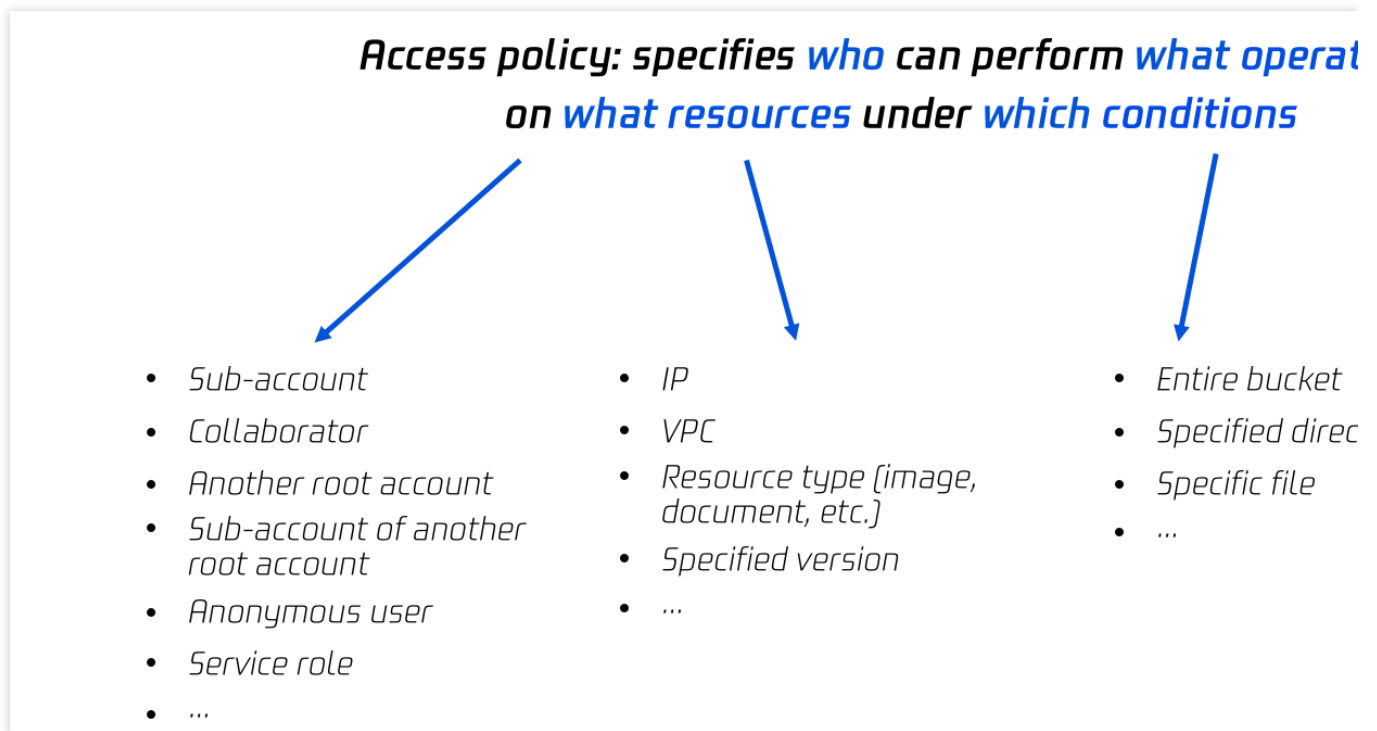
COS resources (buckets and objects) are private by default. Only Tencent Cloud root accounts (resource owners) can access and modify them. Other unauthorized users (such as sub-accounts and anonymous users) cannot access objects by using URLs.

After creating a Tencent Cloud sub-account, you can configure an access policy to authorize it. If you want to open up resources (buckets, objects, and directories) to non-Tencent Cloud users, you can set the permissions of the resources to public read.



Access Control Elements

You can grant access permissions by specifying a user to perform a specified action on specified resources under a specified condition. Generally, the following four elements are used to describe an access policy: **principal**, **resource**, **action**, and **condition (optional)**.



Access Policy Elements

Tencent Cloud identity (principal)

When you sign up for a Tencent Cloud account, the system creates a root account identity for you to log in to Tencent Cloud services. With the root account, you can use the user management feature to manage different user types, such as **collaborator**, **message recipient**, **sub-user**, and **role**. For more information, see [User Types](#) and [Glossary](#) of CAM.

Note:

Suppose you want to authorize a colleague, you first need to create a sub-user in the [CAM console](#), select a [bucket policy](#), [ACL](#), and/or [user policy](#) to set specific permissions for the sub-user.

COS resources

Buckets and objects are basic resources of the COS service. Folders are a special type of object. You can authorize objects in a folder by authorizing the folder. For more information, see [Setting Folder Permissions](#).

Buckets and objects have subresources associated with them.

Subresources associated with a bucket include:

acl and policy: Access control list of a bucket

website: Static website hosting configuration of a bucket

tagging: Tag information of a bucket

cors: Cross-origin resource sharing (CORS) configuration of a bucket

lifecycle: Lifecycle configuration of a bucket

Subresources associated with an object include:

acl: Access control information of an object

restore: Restoration configuration of an archived object

COS operations (actions)

COS provides a range of API operations on various resources. For more information, see [Operation List](#).

COS conditions (optional)

COS conditions refer to conditions for permissions to take effect, such as VPC and VIP. For more information, see [Conditions](#).

Private Principle

Note:

COS resources are private by default.

The resource owner (the Tencent Cloud root account creating a bucket resource) has the highest permission on the resource, and can grant access permissions to others or anonymous users by writing or modifying an access policy.

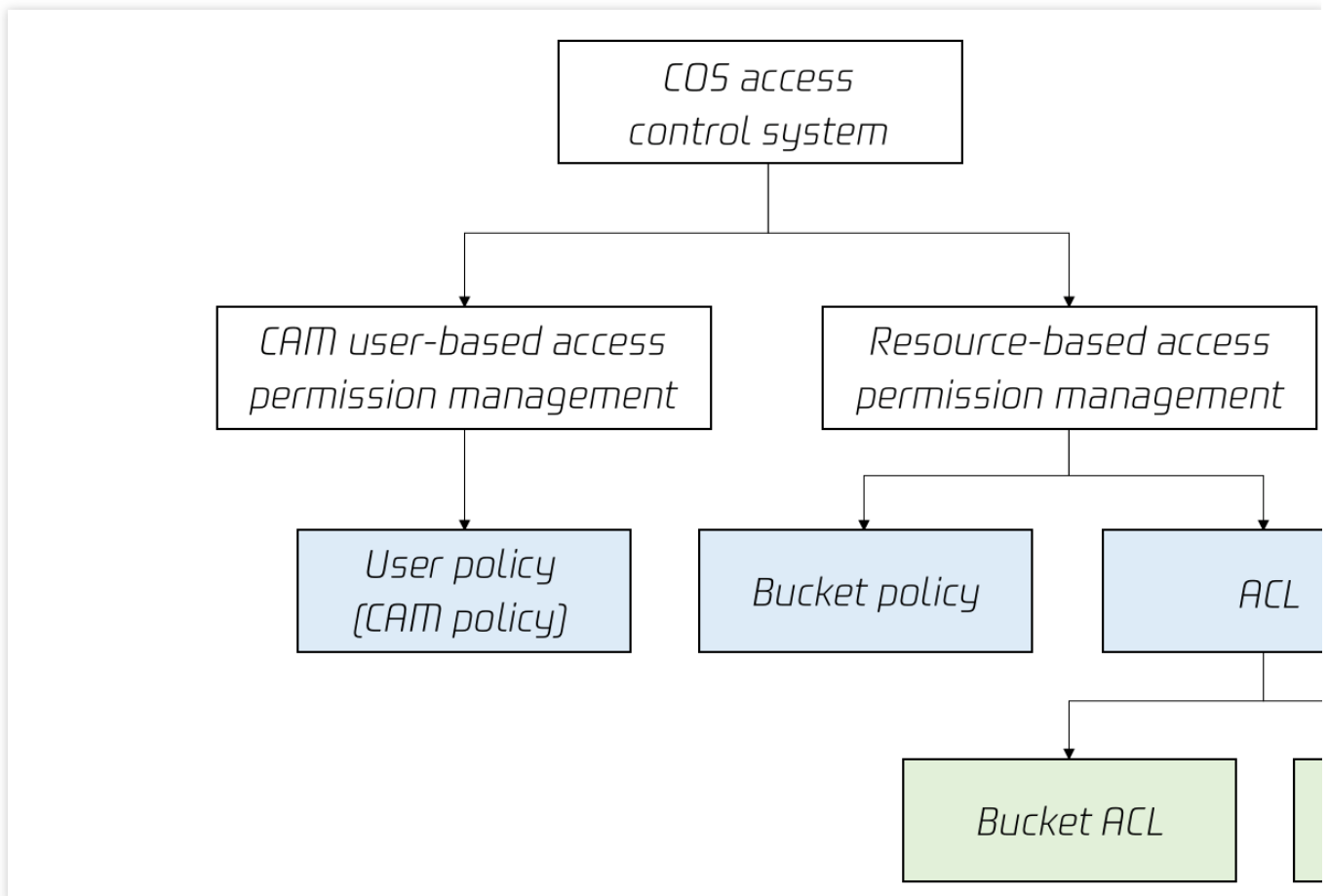
When a [CAM](#) account is used to create a bucket or upload an object, its parent root account is the resource owner.

The root account of a bucket owner can authorize other Tencent Cloud root accounts to upload objects (i.e. cross-account upload). In this case, the object owner is still the root account of the bucket owner.

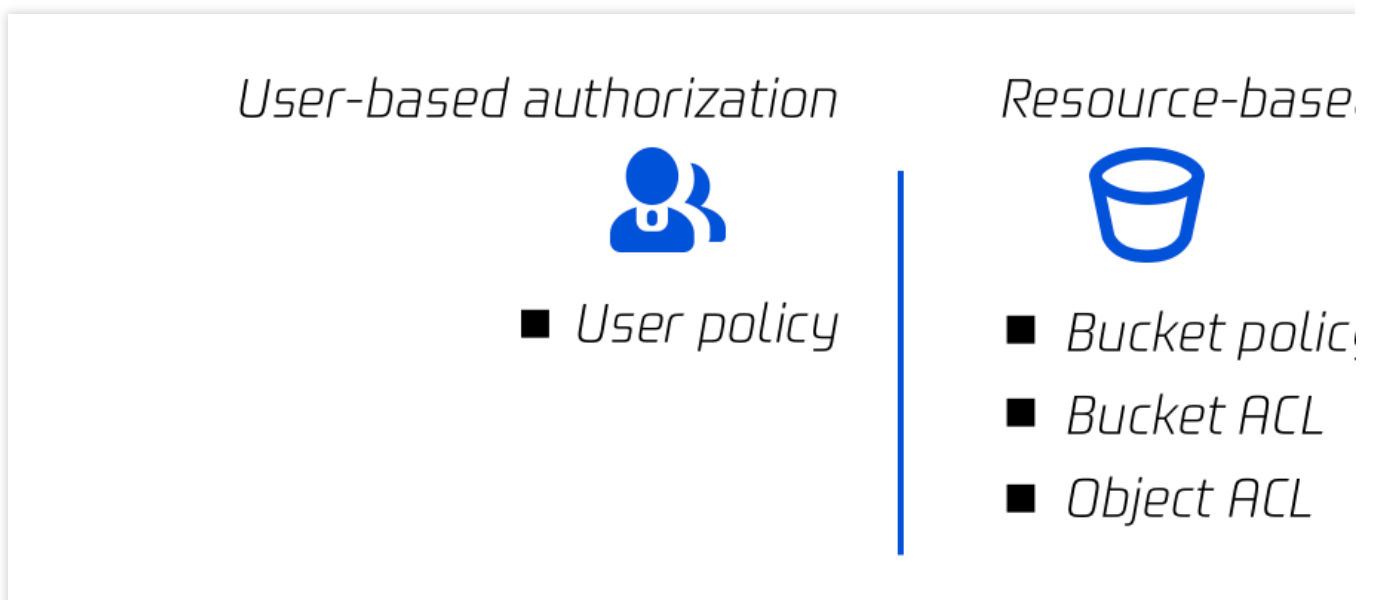
Access Control Methods

COS provides multiple permission setting methods to implement access control, including bucket policy, user policy (CAM policy), bucket ACL, and object ACL.

These methods can be categorized into resource-based authorization and user-based authorization based on the starting point of policy setting, or categorized into policy-based authorization and ACL-based authorization based on the authorization mode.



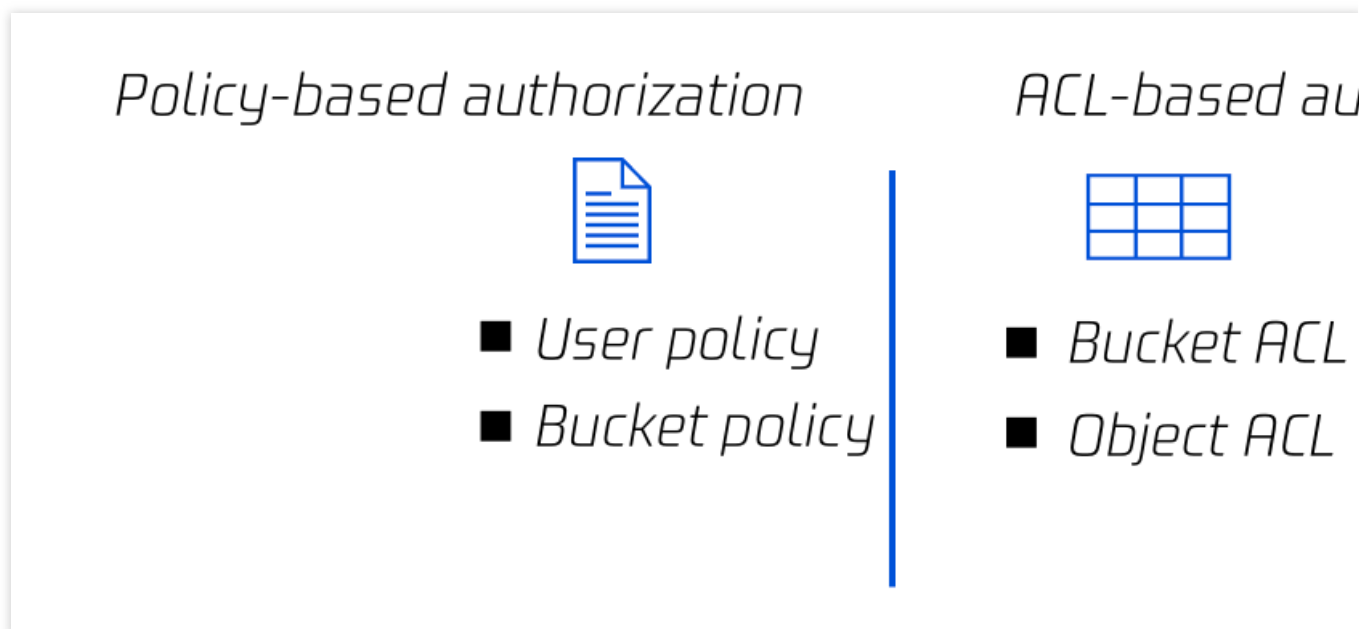
Categorization option 1: Resource-based authorization vs. user-based authorization



Resource-based authorization: Resource policies associate permissions with specific resources, including bucket policies, bucket ACLs, and object ACLs. Resource policies can be configured in the COS console or through APIs.

User-based authorization: User policies (CAM policies) associate permissions with users. When configuring a user-based policy, you don't need to specify users. Instead, you only need to specify resources, actions, conditions, and so on. User policies can be configured in the [CAM console](#).

Categorization option 2: Policy-based authorization vs. ACL-based authorization



Policy-based authorization: Both user policies (CAM policies) and bucket policies implement authorization based on complete policy syntax. They authorize refined actions specific to each API and allow specifying the `allow` or `deny` effect.

ACL-based authorization: Bucket ACLs and object ACLs are implemented based on ACLs. An ACL is a list of specified grantees and granted permissions, which is associated with resources and corresponds to organized and abstracted permissions. ACLs allow specifying only the `allow` effect.

Resource-based policy

Resource-based policies are categorized into three types: bucket policy, bucket ACL, and object ACL. COS supports access control at both the **bucket** and **object** levels as detailed below:

Dimension	Type	Language	Supported Identity	Supported Resource Granularity	Supported Action	Supported Effect
Bucket	Access policy language (Policy)	JSON	Sub-accounts, roles, Tencent Cloud services, other root accounts, anonymous users, etc.	Buckets, objects, prefixes, etc.	Each specific action	Allow/Deny
Bucket	Access	XML	Other root accounts,	Buckets	Read and	Allow

	control list (ACL)		sub-accounts, anonymous users, etc.		write actions	
Object	Access control list (ACL)	XML	Other root accounts, sub-accounts, anonymous users, etc.	Objects	Read and write actions	Allow

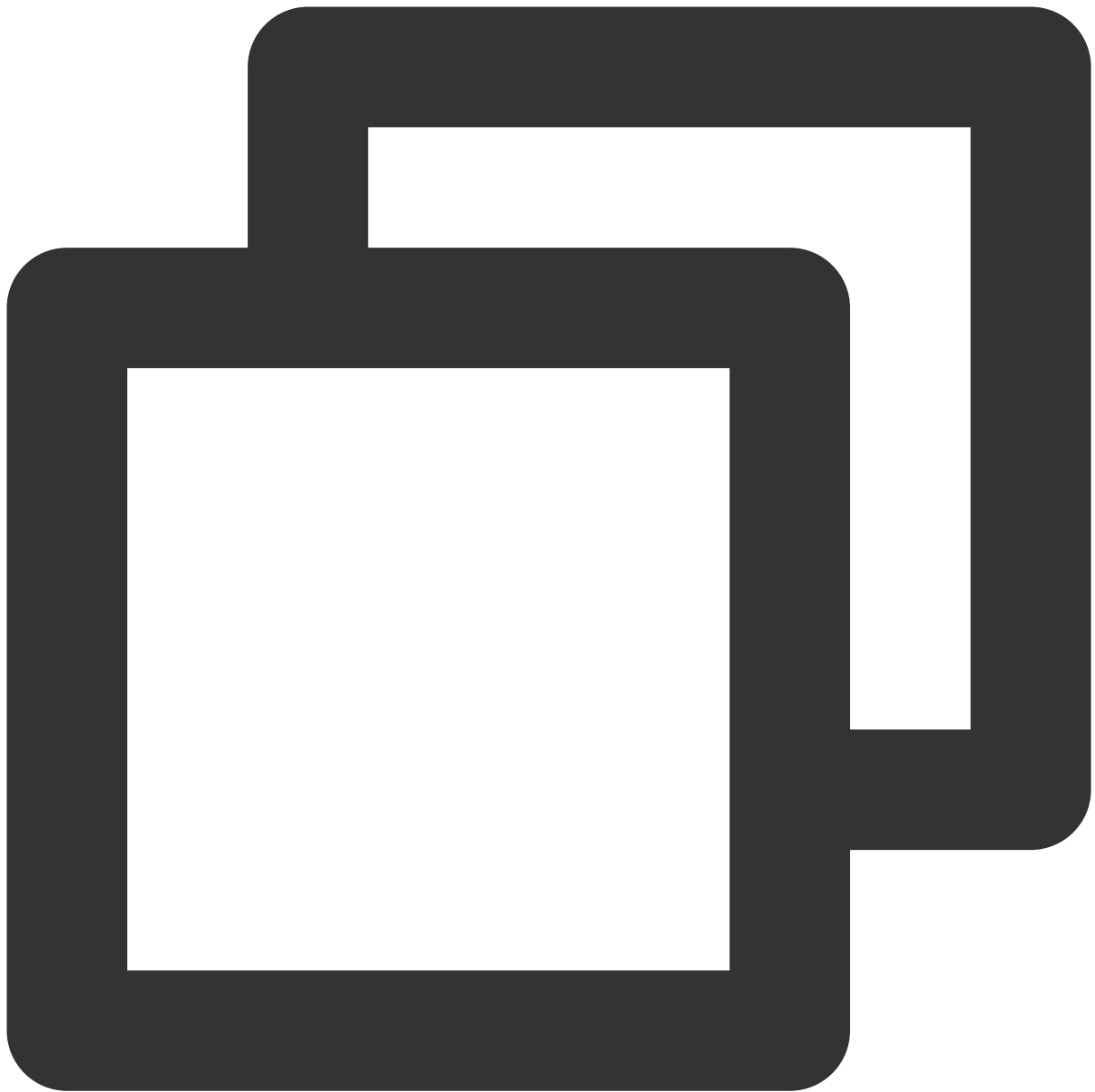
Bucket policy

A bucket policy is described in JSON language and supports granting anonymous identities or any Tencent Cloud [CAM](#) accounts the permissions to access and manipulate buckets and objects. In COS, a bucket policy can be used to manage almost all operations in the bucket. We recommend you use a bucket policy to manage access policies that cannot be described by ACLs. For more information, see [Bucket Policy](#).

Note:

A Tencent Cloud root account has the highest permission on its resources (including buckets). Although you can set limits on almost all operations in a bucket policy, the root account always has the permission for the `PUT Bucket Policy` operation and can call this operation without checking the bucket policy.

The following policy allows anonymous users to access all objects in the bucket `examplebucket-1250000000` in Guangzhou and to download all objects (via `GetObject`) in the bucket without signature verification. In this case, any anonymous user who knows the URLs can download the objects (similar to public read):



```
{
  "Statement": [
    {
      "Principal": "*",
      "Effect": "Allow",
      "Action": ["cos:GetObject"],
      "Resource": ["qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"]
    }
  ],
  "Version": "2.0"
}
```

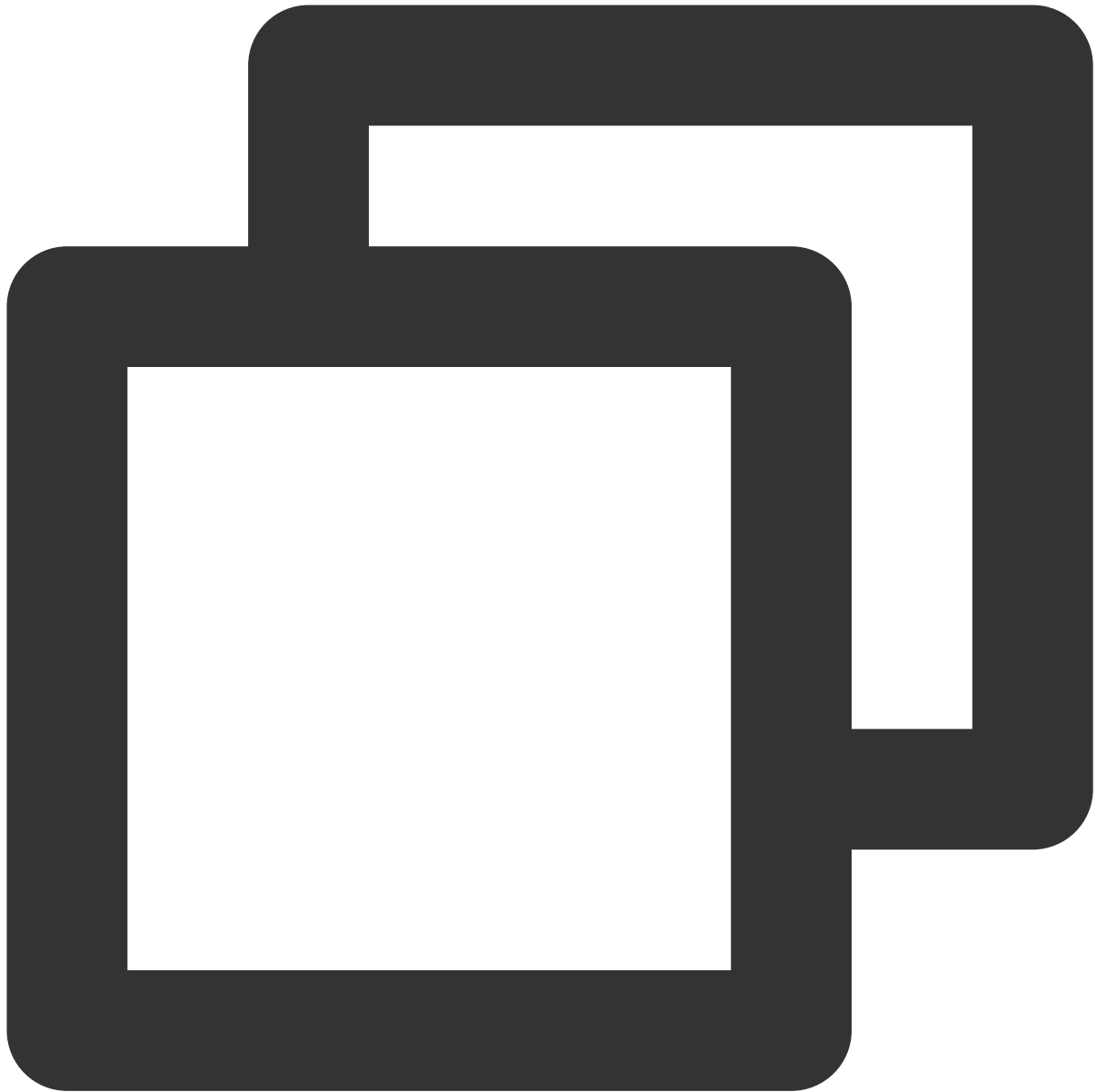
ACL

An ACL is described in XML language. It is a list of specified grantees and granted permissions. Each bucket or object has an associated ACL, which allows granting basis read/write permissions to anonymous users or other Tencent Cloud root accounts. For more information, see [ACL](#).

Note:

The resource owner always has the FULL_CONTROL permission on the resource, regardless of whether this is described in the distributed ACL.

The bucket ACL in the following example describes the full control permission of the bucket owner (UIN: 1000000000001):

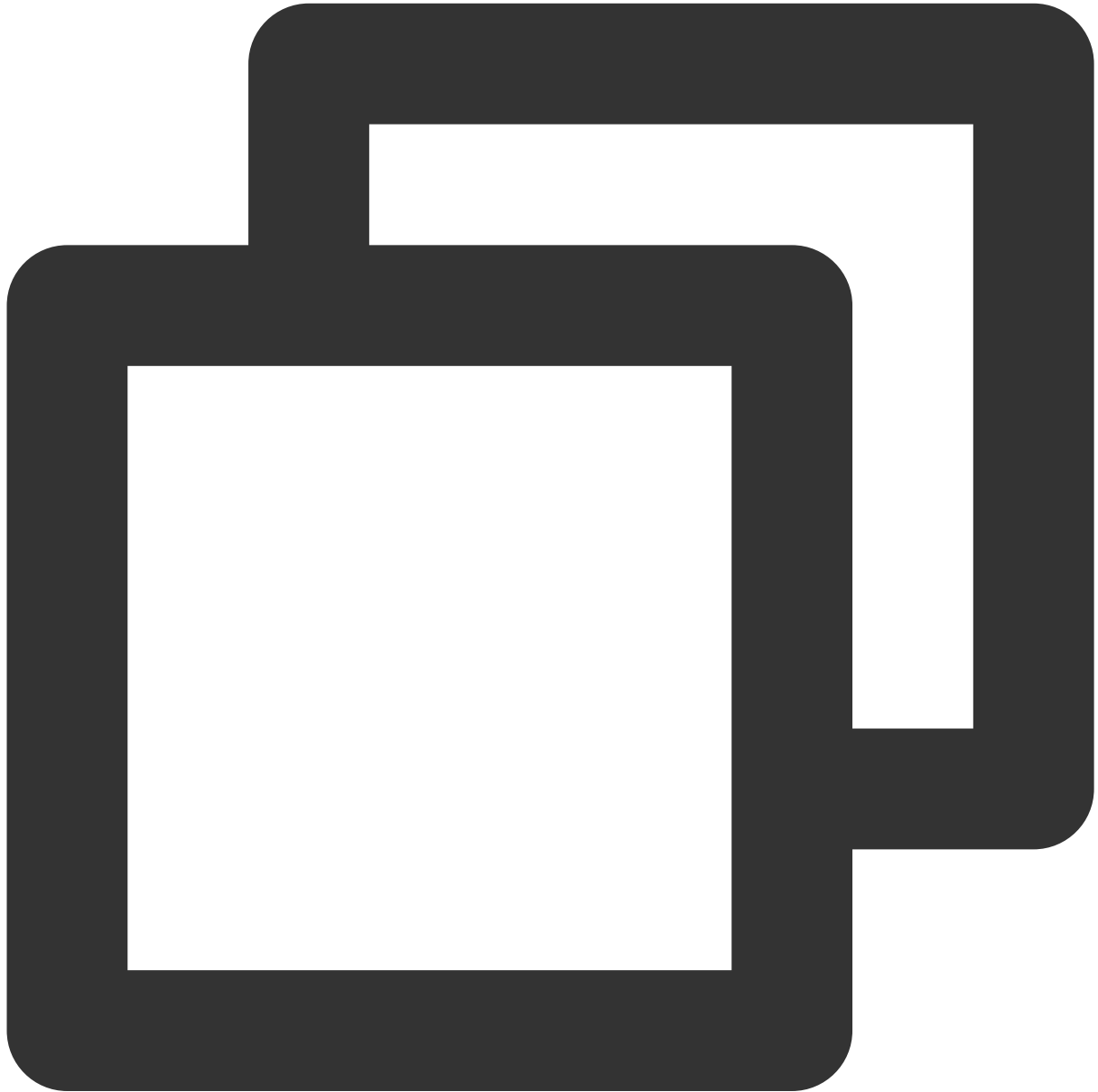


```
<AccessControlPolicy>
  <Owner>
    <ID>qcs::cam::uin/1000000000001:uin/1000000000001</ID>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Root"
        <ID>qcs::cam::uin/1000000000001:uin/1000000000001</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
```



```
</AccessControlList>  
</AccessControlPolicy>
```

The object ACL in the following example describes the full control permission of the object owner (UIN: 1000000000001) and grants the read permission to all users (the public-read permission to anonymous users):



```
<AccessControlPolicy>  
  <Owner>  
    <ID>qcs::cam::uin/1000000000001:uin/1000000000001</ID>  
  </Owner>  
  <AccessControlList>
```

```
<Grant>
  <Grantee>
    <ID>qcs::cam::uin/1000000000001:uin/1000000000001</ID>
  </Grantee>
  <Permission>FULL_CONTROL</Permission>
</Grant>
<Grant>
  <Grantee>
    <URI>http://cam.qcloud.com/groups/global/AllUsers</URI>
  </Grantee>
  <Permission>READ</Permission>
</Grant>
</AccessControlList>
</AccessControlPolicy>
```

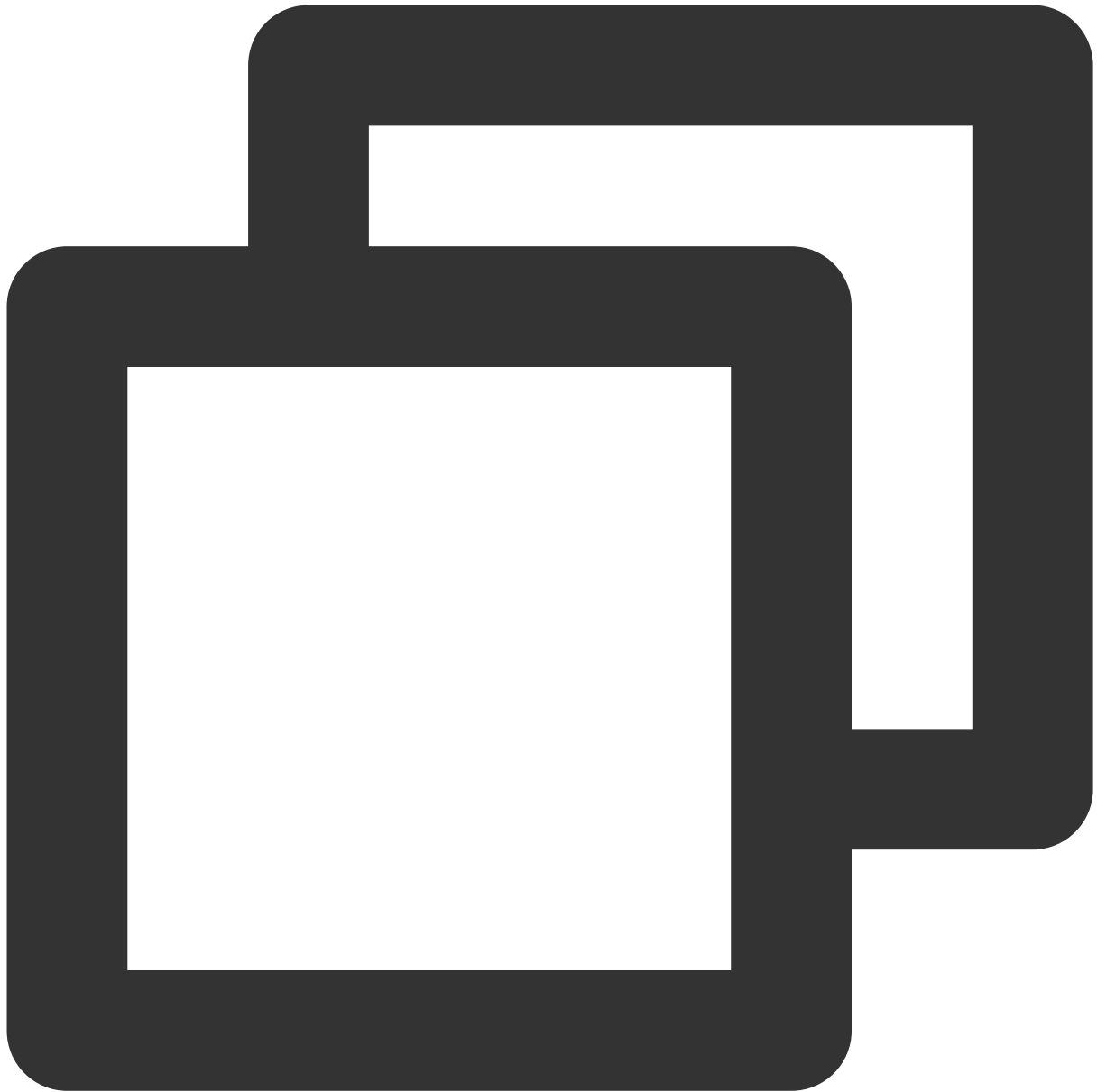
User policy

In [CAM](#), you can grant different permissions to different types of users under a root account.

The biggest difference between a user policy and a bucket policy is that the former only describes the effect, action, resource, and condition (optional) but not the principal. Therefore, you have to write a user policy first, and then associate it manually with a sub-user, user group, or role. Besides, a user policy cannot grant anonymous users access to resources or operations.

You can associate a preset policy for authorization as described in [Authorization Management](#), or write a user policy by referring to [Element Reference](#) and associate it with a specified identity to manage user access. For more information, see [User Policy](#).

The following sample policy grants the permission to perform all COS operations on the bucket `examplebucket-1250000000` in Guangzhou. You need to save the policy and then associate it with a [CAM](#) sub-user, user group, or role before it can take effect.



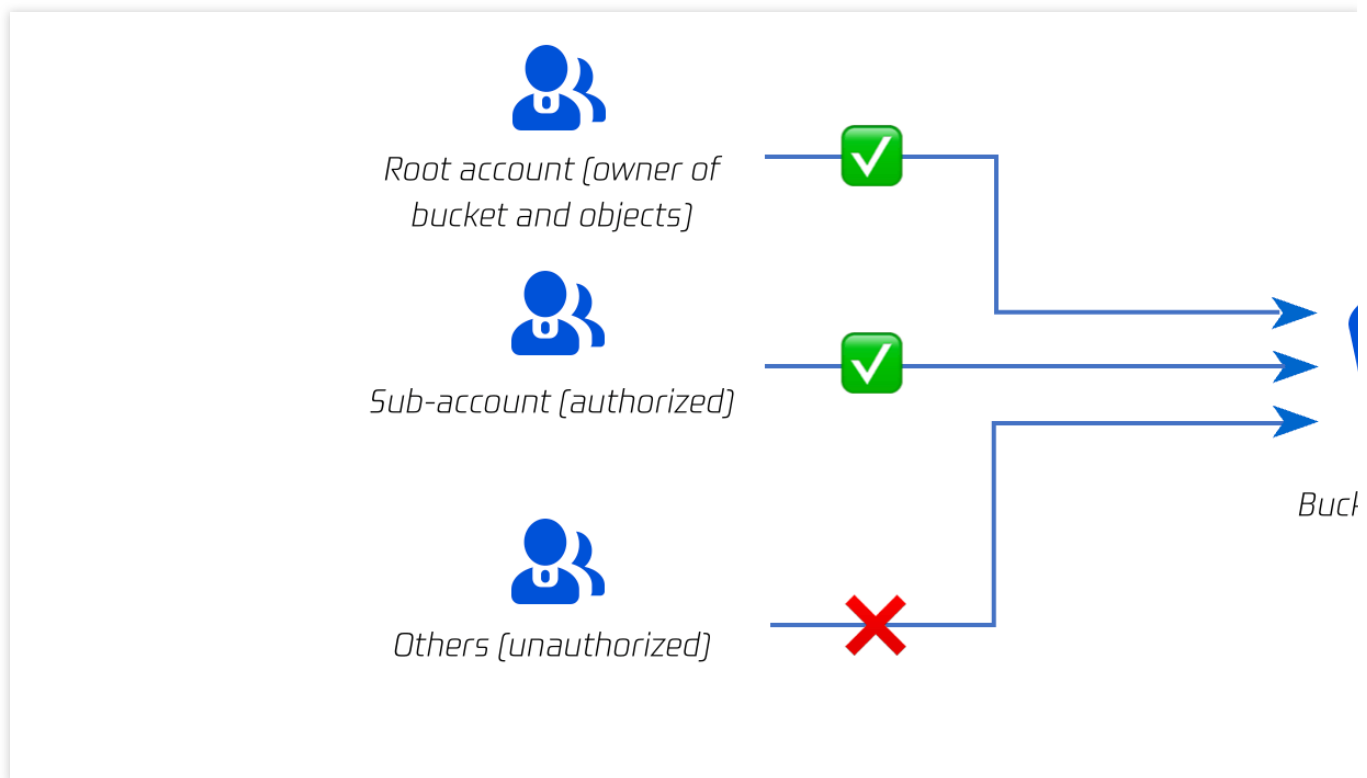
```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["cos:*"],
      "Resource": [
        "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*",
        "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/"
      ]
    }
  ]
},
```

```
"Version": "2.0"  
}
```

COS Authorization and Identity Verification Process

Last updated : 2024-03-25 15:39:58

COS resources (buckets and objects) are configured with the Private Read/Write permission by default. Even if anonymous users obtain object URLs, they cannot access your resource content through the URLs due to lack of signature information.



Main Steps

COS's authorization and identity verification process consists of five steps, as shown in the figure below.

Step 1. Register a Tencent Cloud account



Obtain the root account

Step 2. Activate the COS service



Bucket

Step 3. Create an authorized identity



Root account

Possess all permissions permanently. **No authorization required**



Sub-account

Create a sub-account in CRM. **Authorization required**



Anonymous user. **Authorization required**



Service role, collaborator, etc. **Authorization required**

Step 4. Configure permissions



COS support permissions methods

- User policy
- Bucket policy
- Bucket ACL
- Object ACL

Step 5. Start access

Access methods



Console

Login verification via account and password



http/https

RPI request

Identity verification via keys (SecretId/SecretKey)



Python, Java, ...

SDK

Identity verification via keys (SecretId/SecretKey)

Step 1. Register a Tencent Cloud account

After registration, your account serves as your root account, which has the highest permissions.

Step 2. Activate the COS service

After you activate the COS service, all buckets you create belong to your root account. The root account has the highest permissions of all resources and can be used to create and authorize sub-accounts.

Step 3. Create an authorized identity

Note:

Unless you set the bucket or object permission to Public Read, identity verification is required for any access to COS.

With a root account, you can create multiple identities and grant different permissions to different resources.

If you need to authorize specific users, such as coworkers and users in specific departments, you can create sub-accounts for these users in the [Cloud Access Management \(CAM\) console](#), and then use various authorization methods, such as bucket policies, user policies (CAM policies), bucket ACLs, and object ACLs, to grant specific access permissions to specified resources to the sub-accounts.

If you need to authorize anonymous users, such as allowing others to download objects through URLs without identity verification, you need to change the resource permission from the default Private Read to Public Read.

If you need to enable other Tencent Cloud services (such as CDN) to use COS buckets, you need to follow the same authorization process. With your authorization, these services will legally access COS through service roles. You can view the created service roles in the CAM console.

For authorization across Tencent Cloud accounts, if you need to authorize only one COS bucket, you can directly authorize another root account via a bucket policy or ACL. If you need to authorize multiple COS buckets or Tencent Cloud resources, you can create collaborator identities in the [CAM console](#) for a wider range of authorization.

Step 4. Configure permissions for the identity

COS supports various permission configuration modes, including [bucket policies](#), [user policies \(CAM policies\)](#), [bucket ACLs](#), and [object ACLs](#). You can choose an authorization mode according to your use case.

Step 5. Start access and identity verification

You can access COS through the console, API requests, and SDKs. For security reasons, buckets are configured with the Private Read permission by default and require identity verification for any access mode. For the console access mode, you can log in using your account and password. For both the API request and SDK access modes, you need to use keys (SecretId/SecretKey) for identity verification.

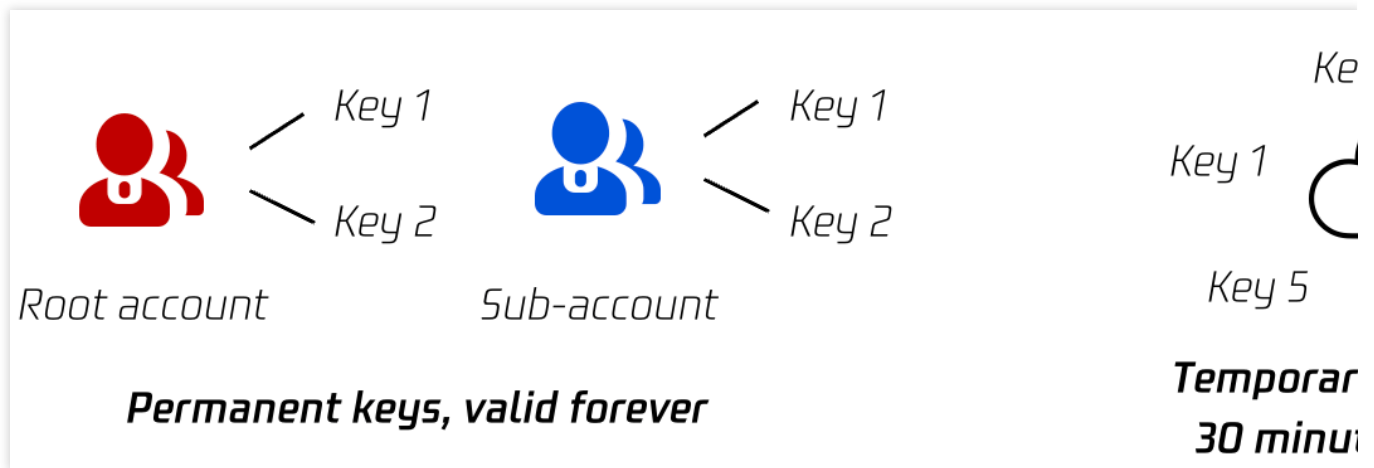
COS Identity Verification Modes

Identity verification*Access via permanent keys**SecretId
SecretKey**Forever**Access via temporary keys**SecretId
SecretKey
Token**30 minutes**Access via temporary URL
[pre-signed URL]**Object URL**<https://test-12345678.cos.ap-beijing.myqcloud.com/test.psign-time=1638417770;1638421370&q-key-time=1638417770;1638421370&q-signature=xxxxxx6&x-cos-security-token=xxxxxx6>**Anonymous access**Object URL**<https://test-12345678.cos.ap-beijing.myqcloud.com/test>*

COS buckets are private by default. Identity verification is required regardless of whether you access COS using keys (permanent or temporary keys) or pre-signed URLs. In special scenarios, you can set your bucket permission to Public Read so that any user can use object URLs to download objects directly without identity verification, which is risky.

1. Access via permanent key

A key (SecretId and SecretKey) is the security credential used for identity verification when a user accesses a Tencent Cloud API and can be viewed on the [Manage API Key](#) page in the CAM console. Multiple keys can be created under each root account or sub-account.



A permanent key consists of a SecretId and a SecretKey. Two pairs of permanent keys can be generated for each root account or sub-account. Permanent keys represent the permanent identities of accounts. They are valid forever if not deleted. For more information, see [Accessing COS Using a Temporary Key](#).

2. Access via temporary key

A temporary key consists of a SecretId, SecretKey, and token. Multiple temporary keys can be generated for each root account or sub-account. Compared with permanent keys, temporary keys have shorter validity periods (1,800 seconds by default). The validity period of a temporary key can be up to 7,200 seconds for a root account and up to 129,600 seconds for a sub-account. For more information, see [Getting Temporary Credentials for a Federated User](#). Temporary keys are applicable to temporary authorization scenarios such as frontend direct upload. Compared with permanent keys, distributing temporary keys to untrusted users is more secure. For more information, see [Accessing COS Using a Temporary Key](#).

3. Access via temporary URL (pre-signed URL)

For more information, see [Accessing COS via Pre-Signed URL](#).

Downloading an object

If you want any third party to be able to download an object from your bucket, but you don't want them to use CAM accounts or temporary keys, signatures can be provided by pre-signed URLs for temporary download operations. Anyone who receives a valid pre-signed URL can download an object.

Obtaining a temporary download link (validity period: 1-2 hours) from the COS console or COSBrowser

You can obtain the temporary download link of an object from the COS console or COSBrowser and enter the link in the browser to download the object. For more information, see [Obtaining a Temporary Link Quickly](#).

Generating a pre-signed URL via SDK

You can use an SDK to obtain pre-signed URLs with custom validity periods in batches. For more information, see [Obtaining Pre-Signed URLs in Batches via SDK](#).

Generating a pre-signed URL via signature generation tool

If you are not familiar with programming, you can use a signature generation tool to obtain pre-signed URLs with custom validity periods. For more information, see the signature generation tool documentation.

Manually constructing a pre-signed URL

A pre-signed URL is in fact an object URL concatenated with a signature. Therefore, you can use an SDK or signature generation tool to generate a signature and concatenate the object URL and the signature to form a pre-signed URL. However, this method is generally not recommended because of the complexity of the signature generation algorithm.

Uploading an object

If you want any third party to be able to upload an object to your bucket, but you don't want them to use CAM accounts or temporary keys, signatures can be provided by pre-signed URLs for temporary upload operations. Anyone who receives a valid pre-signed URL can upload an object.

Method 1. Generating a pre-signed URL via SDK

SDKs in various programming languages provide pre-signed URL generation methods. For more information about the methods, see [Upload via Pre-Signed URL](#). Select a method according to the programming language that you are familiar with.

Method 2. Manually constructing a pre-signed URL

A pre-signed URL is in fact an object URL concatenated with a signature. Therefore, you can use an SDK or signature generation tool to generate a signature and concatenate the object URL and the signature to form a pre-signed URL for object upload. However, this method is generally not recommended because of the complexity of the signature generation algorithm.

4. Anonymous access

COS buckets are private by default. Identity verification is required no matter whether you access COS using keys (permanent or temporary keys) or pre-signed URLs.

In special scenarios, you can set your bucket or object permission to Public Read so that any user can use object URLs to download objects directly without identity verification.

Note:

Setting your resource permission to Public Read is risky because, once the resource link is leaked, any user can access your resource, which may cause hotlinking by malicious users.

Setting the permission of a bucket to Public Read

You can set your bucket permission to Public Read in the COS console so that each object in the bucket can be downloaded directly via object URL. For the permission setting method, see [Setting Access Permission](#).

Setting the permission of an object to Public Read

You can set the permission of an object to Public Read in the COS console so that only this object in your bucket can be downloaded directly via object URL, and other objects are not affected. For the permission setting method, see

[Setting Object Access Permission.](#)

Setting the permission of a folder to Public Read

You can set the permission of a folder to Public Read in the COS console so that all objects in the folder can be downloaded directly via object URL, and objects outside of the folder are not affected. For the permission setting method, see [Setting Folder Permissions](#).

Notes on Principle of Least Privilege

Last updated : 2024-03-25 15:33:39

Overview

When using COS, you may need to use a temporary key to grant users permissions to certain resources or operations, configure user policies for your sub-users or collaborators that allow them to help you operate on the resources in COS, or create bucket policies that allow the specified users to perform certain operations on or access certain resources in your bucket. When configuring these permissions, please comply with the **principle of least privilege** in order to ensure the security of your data assets.

The **principle of least privilege** means that when granting permission, you must specify the scope of the permission granted to the **specified user** for performing **what operation** and access **what resource** under **what conditions**.

Limits

We recommend you strictly follow the principle of least privilege to ensure that a user can only perform the specified operations (such as `action:GetObject`) or access the specified resources (such as `resource:examplebucket-1250000000/exampleobject.txt`).

To prevent data security risks caused by unexpected and unauthorized operations due to excessive permissions, we strongly recommend you not authorize a user to access all resources (such as `resource:*`) or perform all operations (such as `action:*`).

Below are some potential data security risks:

Data leakage: If you want to authorize a user to download the specified resources such as `examplebucket-1250000000/data/config.json` and `examplebucket-1250000000/video/` but include `examplebucket-1250000000/*` in the permission policy, then all objects in the bucket can be downloaded without your authorization, leading to unexpected data leakage.

Data overwriting: If you want to authorize a user to upload `examplebucket-1250000000/data/config.json` and `examplebucket-1250000000/video/` but include `examplebucket-1250000000/*` in the permission policy, then all objects in the bucket can be uploaded without your authorization, which may overwrite unintended objects. To avoid this risk, in addition to following the principle of least privilege, you can retain all versions of data for traceability as instructed in [Overview](#).

Permission leakage: If you want to authorize a user to list the objects in the bucket (`cos:GetBucket`) but configured `cos:*` in the permission policy, then all operations on the bucket will be allowed, including reauthorizing the bucket, deleting objects, and deleting the bucket, which puts your data at extremely high risk.

Usage Guide

Under the principle of least privilege, you should specify the following information in the policy:

principal: you should specify to which sub-account (user ID required), collaborator (user ID required), anonymous user, or user group to grant permission. This is not needed if you use a temporary key for access.

statement: enter the corresponding parameters.

effect: you must specify whether the policy is to "allow" or "deny".

action: you must specify the action to allow or deny. It can be one API operation or a set of API operations.

resource: You must specify the resource for which permission is granted. A resource is described in a six-segment format. You can set the resource as a specific file, e.g., `exampleobject.jpg` or a directory, e.g.,

`examplePrefix/*`. Unless needed, do not grant any user the access to all of your resources using the `*` wildcard.

condition: it describes the condition for the policy to take effect. A condition consists of operator, action key, and action value. A condition value may contain information such as time and IP address.

Least privilege guide for temporary keys

When applying for a temporary key, you can set the `policy` field as described in [GetFederationToken](#) to grant limited permissions on operations and resources. For more information about how to generate a temporary key, see [Generating and Using Temporary Keys](#).

Authorization example

Granting a user permission to access the specified object using the SDK for Java

If you want to use the Java SDK to grant a user permission to download the `exampleObject.txt` object in the `examplebucket-1250000000` bucket, the configuration code should be as follows:



```
// Import `java sts sdk` using the integration method with Maven as described on Gi
import java.util.*;
import org.json.JSONObject;
import com.tencent.cloud.CosStsClient;

public class Demo {
    public static void main(String[] args) {
        TreeMap<String, Object> config = new TreeMap<String, Object>();

        try {
            String secretId = System.getenv("secretId");// User `SecretId`. We reco
```

```
String secretKey = System.getenv("secretKey");// User `SecretKey`. We r
// Replace it with your own SecretId
config.put("SecretId", secretId);
// Replace it with your own SecretKey
config.put("SecretKey", secretKey);

// Validity period of the temporary key, in seconds. Default value: 1800
config.put("durationSeconds", 1800);

// Replace it with your own bucket
config.put("bucket", "examplebucket-1250000000");
// Replace it with the region where your bucket resides
config.put("region", "ap-guangzhou");

// Change it to the allowed path prefix (such as "a.jpg", "a/*", or "**")
// If "*" is entered, you allow the user to access all resources. Unless
config.put("allowPrefix", "exampleObject.txt");

// List of key permissions. The following permissions are required for
String[] allowActions = new String[] {
    // Download data
    "name/cos:GetObject"
};
config.put("allowActions", allowActions);

JSONObject credential = CosStsClient.getCredential(config);
// If it succeeds, the temporary key information will be returned and p
System.out.println(credential);
} catch (Exception e){
    // If it fails, an exception will be thrown
    throw new IllegalArgumentException("no valid secret !");
}
}
```

Granting a user permission to access the specified object using API

If you want to use an API to grant a user permission to download the `exampleObject.txt` object in the `examplebucket-1250000000` bucket and all objects in the `examplePrefix` directory, the access policy should be as follows:



```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:GetObject"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-beijing:uid/1250000000:examplebucket-1250000000/exampleObject.",
        "qcs::cos:ap-beijing:uid/1250000000:examplebucket-1250000000/examplePrefix/"
      ]
    }
  ]
}
```



```
]
}
]
}
```

Least privilege guide for signatures

You can perform temporary uploads and downloads using pre-signed URLs. Moreover, if you send a valid pre-signed URL to others, he (or she) can upload or download the objects.

Note:

Both temporary and permanent keys can be used to generate pre-signed URLs. However, you are advised to follow the least privilege principle when [generating a temporary key](#) and use the temporary key to calculate the signature. Try to avoid using a permanent key that has excessive permissions for the sake of security.

Authorization example

Granting a user permission to use a pre-signed URL to download an object

Use a temporary key to generate a signed download URL and set it to overwrite some public headers to be returned (such as `content-type` and `content-language`). The Java code sample is as follows:



```
// Pass in the obtained temporary key (tmpSecretId, tmpSecretKey, sessionToken)
String tmpSecretId = "SECRETID";
String tmpSecretKey = "SECRETKEY";
String sessionToken = "TOKEN";
COSCredentials cred = new BasicSessionCredentials(tmpSecretId, tmpSecretKey, sessionToken);
// Set the bucket region. For abbreviations of COS regions, see https://cloud.tencent.com/document/product/436/14074
// `clientConfig` contains the set methods to set region, HTTPS (HTTP by default),
Region region = new Region("COS_REGION");
ClientConfig clientConfig = new ClientConfig(region);
// To generate a URL that uses the HTTPS protocol, configure this line (recommended)
// clientConfig.setHttpProtocol(HttpProtocol.https);
```

```
// Generate a COS client.
COSClient cosClient = new COSClient(cred, clientConfig);
// Enter the Bucket name in the format of `BucketName-APPID`.
String bucketName = "examplebucket-1250000000";
// Object key, the unique identifier of the object in the bucket.
String key = "exampleobject";
GeneratePresignedUrlRequest req =
    new GeneratePresignedUrlRequest(bucketName, key, HttpMethodName.GET);
// Set the http header returned for download.
ResponseHeaderOverrides responseHeaders = new ResponseHeaderOverrides();
String responseContentType = "image/x-icon";
String responseContentLanguage = "zh-CN";
// Set the returned header to contain filename information.
String responseContentDisposition = "filename=\\\"exampleobject\\\"";
String responseCacheControl = "no-cache";
String cacheExpireStr =
    DateUtils.formatRFC822Date(new Date(System.currentTimeMillis() + 24L * 3600
responseHeaders.setContentType(responseContentType);
responseHeaders.setContentLanguage(responseContentLanguage);
responseHeaders.setContentDisposition(responseContentDisposition);
responseHeaders.setCacheControl(responseCacheControl);
responseHeaders.setExpires(cacheExpireStr);
req.setResponseHeaders(responseHeaders);
// Setting the signature expiration time (optional). If it is not configured, the s
// Set the signature to expire in half an hour.
Date expirationDate = new Date(System.currentTimeMillis() + 30L * 60L * 1000L);
req.setExpiration(expirationDate);
URL url = cosClient.generatePresignedUrl(req);
System.out.println(url.toString());
cosClient.shutdown();
```

Least privilege guide for user policy

A user policy is a user permission policy created in the [CAM Console](#) to grant a user permission to access certain resources in COS. For more information, please see [Access Policy Language Overview](#).

Authorization example

Granting an account permission to access the specified object

If you want to grant an account whose UIN is `100000000001` permission to download the

`exampleObject.txt` object in the `examplebucket-1250000000` bucket, the access policy should be as follows:



```
{
  "version": "2.0",
  "principal": {
    "qcs": [
      "qcs::cam::uin/1000000000001:uin/1000000000001"
    ]
  },
  "statement": [
    {
      "action": [
        "name/cos:GetObject"
      ]
    }
  ]
}
```

```
    ],
    "effect": "allow",
    "resource": [
        "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000.ap-g
    ]
}
]
```

Granting a sub-account permission to access the specified directory

If you want to grant a sub-account whose UIN is `1000000000011` (root account UIN: `1000000000001`) permission to download the objects in the `examplePrefix` directory in the `examplebucket-1250000000` bucket, the access policy should be as follows:



```
{
  "version": "2.0",
  "principal": {
    "qcs": [
      "qcs::cam::uin/1000000000001:uin/1000000000011"
    ]
  },
  "statement": [
    {
      "action": [
        "name/cos:GetObject"
      ]
    }
  ]
}
```

```
    ],
    "effect": "allow",
    "resource": [
        "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000.ap-g
    ]
}
]
```

Least privilege guide for bucket policy

A bucket policy is an access policy configured for a bucket to allow the specified user to perform certain operations on the bucket and resources in it. For more information, please see [Adding Bucket Policies](#).

Authorization example

Granting a sub-account permission to access the specified objects

If you want to grant a sub-account whose UIN is `1000000000011` (root account UIN: `1000000000001`) permission to download the `exampleObject.txt` object in the `examplebucket-1250000000` bucket and all objects in the `examplePrefix` directory, the access policy should be as follows:



```
{
  "Statement": [
    {
      "Action": [
        "name/cos:GetObject"
      ],
      "Effect": "allow",
      "Principal": {
        "qcs": [
          "qcs::cam::uin/1000000000001:uin/1000000000011"
        ]
      }
    }
  ]
}
```



```
    },  
    "Resource": [  
      "qcs::cos:ap-beijing:uid/1250000000:examplebucket-1250000000/exampleObject.",  
      "qcs::cos:ap-beijing:uid/1250000000:examplebucket-1250000000/examplePrefix/"  
    ]  
  },  
  ],  
  "version": "2.0"  
}
```

Access Policy Evaluation Process

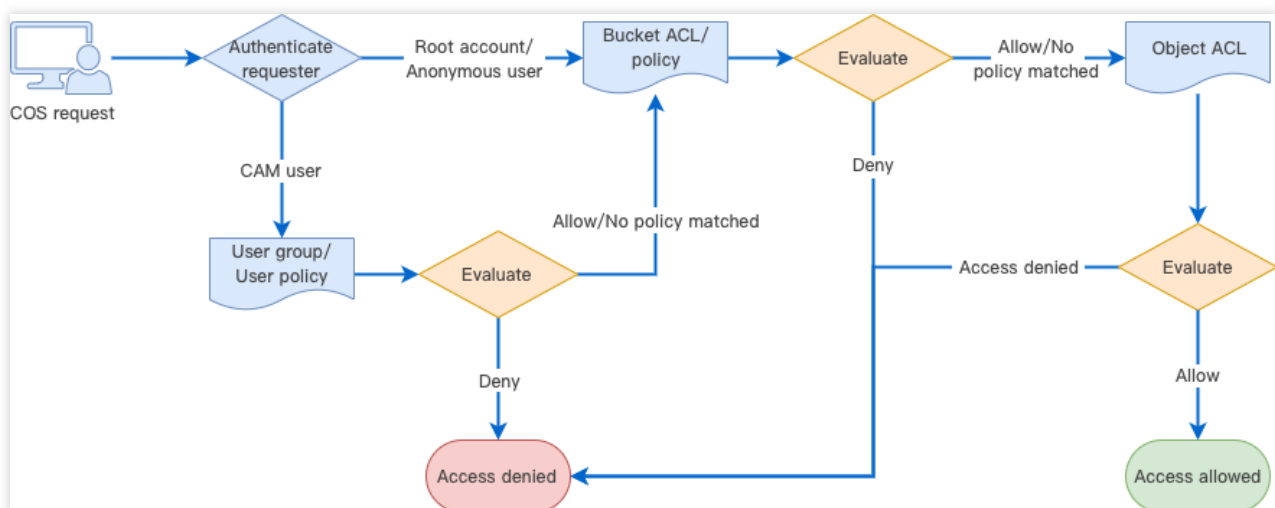
Last updated : 2024-03-25 15:33:39

COS buckets and resources are accessible only to users who have permission. In Tencent Cloud, a root account, by default, holds all management permissions for COS buckets and resources it owns. Any CAM users (other root accounts, collaborators and sub-accounts), anonymous or other type of users should get permission from the root account first before access.

Access policies in COS include user group policies, user policies, bucket access control lists (ACLs), and bucket policies. As shown below, the evaluation of access policies depends on three key factors:

1. User authentication: when a user tries to access a resource in COS, either of the following two procedures applies: If it is a signed request, COS parses the user account information out of it, and forwards the request to CAM for identity verification; If it is an unsigned request, the user enters the next step of authentication as anonymous user.
2. Policy type: access policies in COS include user group policies, user policies, bucket policies and bucket ACLs. The policy type determines the policy priority.
3. Policy context: whether a resource access request will succeed is finally determined based on the permission details recorded across the access policies of all types.

Access Policy Evaluation Process



When Tencent Cloud COS receives a request, it first confirms the identity of the requester and verifies that it has necessary permissions, including checking the user policy, bucket access policy, and resource-based ACL, and authenticating the request.

When COS receives a request, it first performs identity verification, the result of which determines the requester type. Different types may result in different actions:

1. Verified Tencent Cloud root account: a root account holds all operation permissions for the resources that it owns. For resources beyond those, the resource permissions should be evaluated, and the resource access will be permitted only upon successful authentication.
2. Verified CAM user (sub-account or collaborator): needs evaluation of user policies. A CAM user must be authorized by its parent root account to be allowed to initiate an access. To access a resource in another root account, the CAM user should be evaluated for the resource permissions of its root account, and the resource access will be granted only upon successful authentication.
3. Unidentified anonymous user: needs evaluation of resource permissions, including permissions to access the policies or ACLs of buckets and objects in a bucket. The resource access will be granted upon successful authentication.
4. Access will be denied for any requesters other than the above types of users.

How Access Policy Evaluation Works

In the Tencent Cloud permissions system, the access policy evaluation always occurs based on the policy context in the following basic principles:

1. By default, all requests are implicitly denied while a root account enjoys access permissions for all resources that it owns.
2. If an explicit allow exists in the user group/user/bucket policy, or bucket/object ACL, the above default is overridden.
3. An explicit deny overrides any allows in any policy.
4. The validity scope of permissions depends on the union of identity-based policies (user group/user policy) and resource-based policies (bucket policy or bucket/object ACL).

Note:

Explicit deny: is specified for certain users in the user/user group/bucket policy. For example, if a root account configures an explicit deny in its user policy targeted at a sub-user UIN 100000000011 who attempts to `GET Object`, this sub-user will be unable to download any resources in the root account.

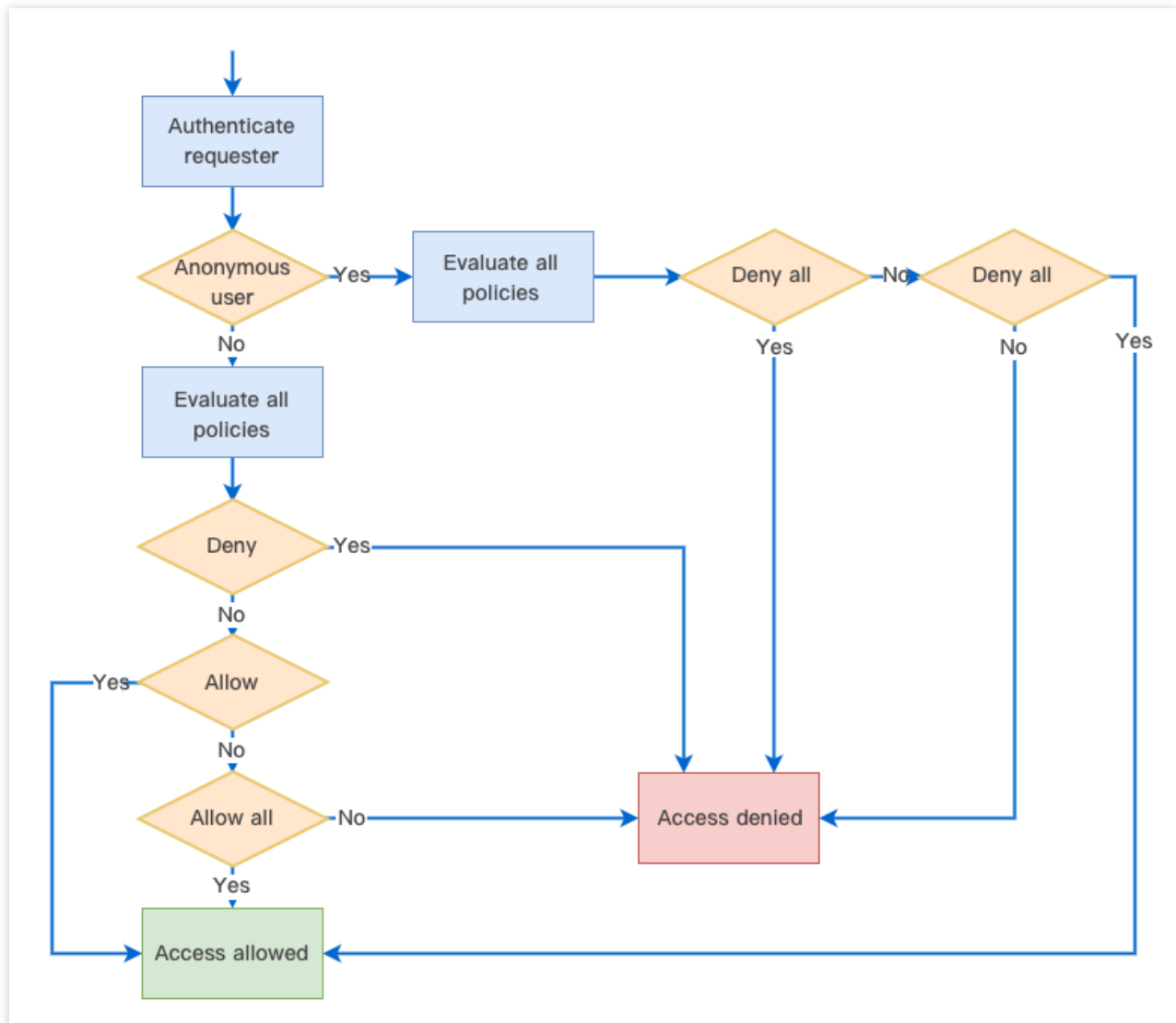
Explicit allow: is specified for certain users in the user/user group/bucket policy or bucket ACL via `grant-*`.

Deny anyone: is specified in the bucket policy. Once it is configured, any unsigned request will be denied while any signed request will be authenticated with identity-based policies.

Allow anyone: is specified in the bucket policy, or `public-*` is specified instead in the bucket ACL.

The validity scope of permissions depends on the union of identity-based and resource-based policies. In a complete authentication, COS first parses the user identity, and then uses it to verify the resources the user has permissions to access. Besides, COS verifies the permissions of the user as anonymous user based on resource-based policies. Either verification is successful and the user will be allowed to access.

The following is a graphical illustration of access policy evaluation. First, evaluate if the requester is an anonymous user based on if the request includes a signature. For an anonymous user, continue to evaluate if “Deny all” or “Allow all” is specified in the policies and determine whether to allow or deny access based on the evaluation result. For a legitimate CAM user or a root account that owns the resource, evaluate if “Deny”, “Allow”, or “Allow all” is specified in the policies, and determine whether to allow or deny access based on the evaluation result.



Policy Context

Policy context provides permission details recorded in your policies. Under the [principle of least privilege](#), you should specify the following in your policies:

principal: you should specify to which sub-account (user ID required), collaborator (user ID required), anonymous user, or user group to grant permission. This is not needed if you use a temporary key for access.

statement: enter the corresponding parameters.

effect: you must explicitly state whether the policy is to "allow" or "deny".

action: you must specify the action to allow or deny. It can be one API operation or a set of API operations.

resource: you must specify the resource for which permission is granted. A resource is described in a six-segment

format. You can set the resource as a specific file, e.g., `exampleobject.jpg` or a directory, e.g.,

`examplePrefix/*`. Unless needed, please do not grant any user the access to all of your resources using the

`*` wildcard.

condition: it describes the condition for the policy to take effect. A condition consists of operator, action key, and action value. A condition value may contain information such as time and IP address.

You should write your policies following a certain policy syntax (See [Access Policy Language Overview](#)) according to your business needs. For examples of writing user and bucket policies, see [Syntax Structure](#) and [Examples of Bucket Policies](#).

Example Access Policy Evaluation

Suppose that a root account UIN 100000000001 associates a preset user policy as detailed below with a sub-account UIN 1000000000011 for it to only read resources in the root account. This policy allows the sub-account to perform all

`List`, `Get`, `Head`, and `OptionsObject` operations.

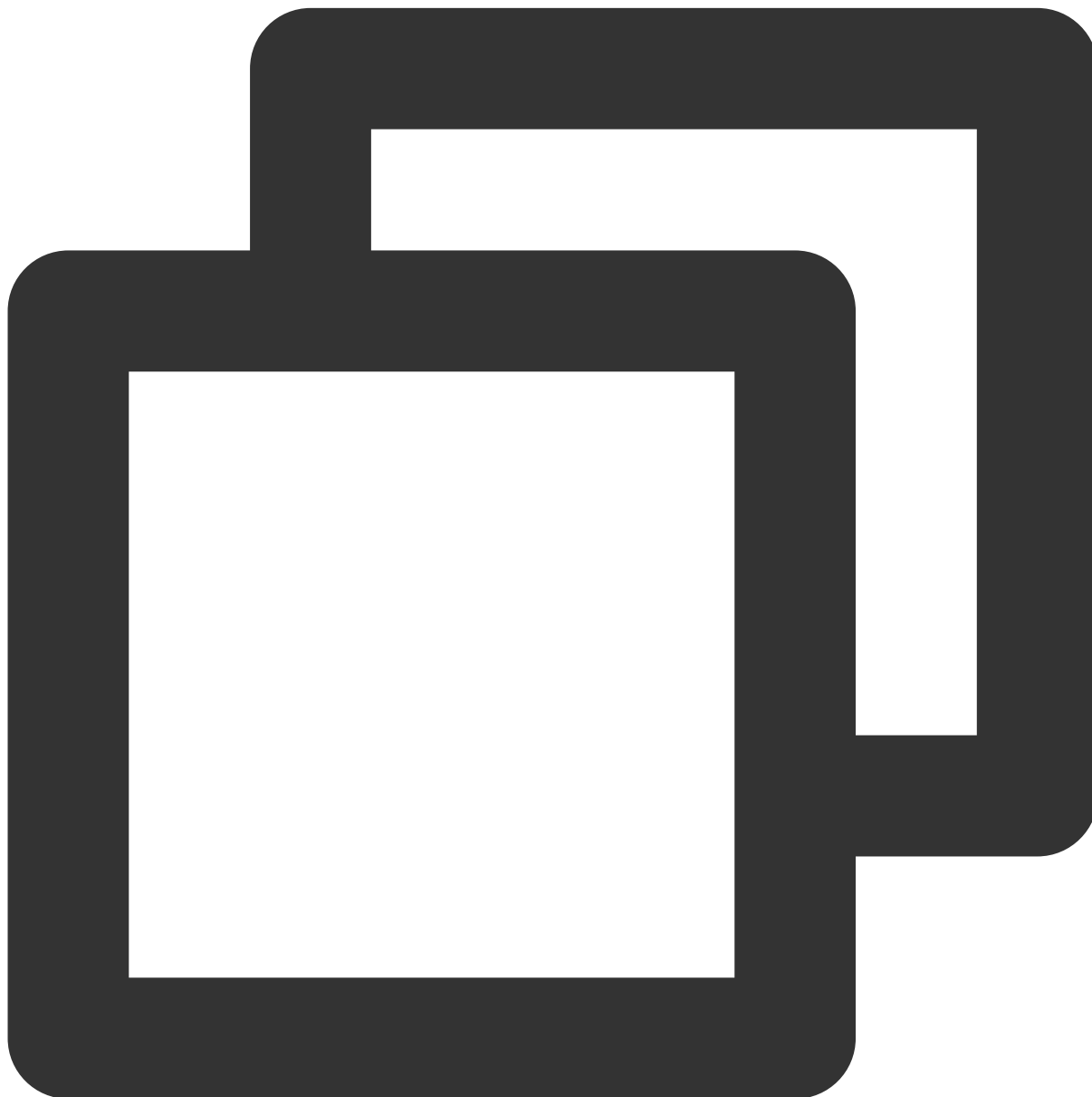


```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "cos:List*",
        "cos:Get*",
        "cos:Head*",
        "cos:OptionsObject"
      ],
      "resource": "*"
    }
  ]
}
```

```
        "effect": "allow"
      }
    ]
  }
```

Also, the root account adds the following bucket policy to a private read/write bucket `examplebucket-`

`1250000000` :



```
{
  "Statement": [
    {
```

```
"Principal": {
  "qcs": [
    "qcs::cam::anyone:anyone"
  ]
},
"Effect": "Deny",
"Action": [
  "name/cos:GetObject"
],
"Resource": [
  "qcs::cos:ap-guangzhou:uid/100000000011:examplebucket-1250000000/*"
]
}
],
"version": "2.0"
}
```

This bucket policy denies any users the ability to download objects (`GetObject`). Therefore, during the access policy evaluation:

1. If the sub-account requests to `GetObject` with a signature, a user policy will be matched based on the requester identity indicated by the request, which will then be granted upon successful access policy evaluation and verification.
2. If the sub-account requests to `GetObject` without a signature, it will be determined automatically as an anonymous requester, and **denied** access by the bucket policy.

Access Control Methods

Bucket Policy

Last updated : 2024-03-25 15:33:39

You can use a bucket policy to grant the operation permissions of a bucket and the objects in the bucket to CAM sub-accounts, other root accounts, and even anonymous users.

Overview

Note:

A Tencent Cloud root account has the highest permission on its resources (including buckets). Even if you can set limits on almost all operations in the bucket policy, the root account always has the permission for the PUT Bucket Policy operation, and can call this operation without checking the bucket policy.

A bucket policy is described in JSON language, and supports granting anonymous identities or any Tencent Cloud [CAM](#) account the permissions to access and perform operations on buckets and objects. In Tencent Cloud COS, the bucket policy can be used to manage almost all operations in the bucket. It is recommended that you use a bucket policy to manage access policies that cannot be described using ACLs.

Application Scenarios

Note:

The permissions for the service-level operations of creating a bucket and querying a bucket list must be configured in the [CAM console](#).

If you want to specify which users can access a COS bucket, you are advised to configure a bucket policy. You can search for a bucket and check the bucket's policy to see who can access the bucket. A bucket policy is recommended in scenarios where you want to:

- Authorize a specific bucket.

- Use higher flexibility than that of an ACL.

- Use cross-account authorization and anonymous user authorization, which are not supported by user policies.

Bucket Policy Elements

A bucket policy is described in JSON language and its syntax complies with the unified specifications of the [access policy language](#). The access policy language contains the following basic elements: principal, effect, action, resource,

and condition. For more information, see [Access Policy Language Overview](#).

The resource scope of a bucket policy is restricted to the bucket, and you can perform authorization on the entire bucket, specified directories, or specified objects.

Note:

When adding an access policy, be sure to grant the minimum permissions needed to satisfy your business needs.

There may be data security risks if you grant other users the permission to access all of your resources

`(resource:*)` or all operations `(action:*)` .

Console Configuration Examples

Note:

When configuring a bucket policy in the COS console, you need to grant users appropriate permissions to the bucket, for example, the permissions to get bucket tags and list bucket permissions.

The bucket policy size is limited to 20 KB.

For example, if you want to grant a sub-account all permissions of a specified directory in a bucket, the corresponding configuration is as follows:

Configuration Item	Description
Effect	Allow
Principal	UIN of the sub-account, which must be a sub-account under the current root account, such as 1000000000011
Resource	Specified directory prefix, such as <code>folder/sub-folder/*</code>
Action	All operations
Condition	None

In the COS console, you can add and manage bucket policies in two modes: [Visual Editor](#) and [JSON](#).

Visual Editor


Click the target bucket and choose **Permission Management > Permission Policy Settings > Visual Editor**. On the **Visual Editor** tab page, click **Add Policy**. In the pop-up window, configure the policy in two steps:

Step 1: select a template (optional)

COS provides you with different templates depending on the combination of authorized users (grantees) and resource scope you choose to help you quickly configure bucket policies. If the templates provided by COS do not meet your

requirements, you can skip this step and add or delete authorized operations in [Step 2: configure the policy](#).

Grantee

All users (allow anonymous access): Select this option if you want to grant operation permissions to anonymous users. If you select this option, all users () will be automatically selected for you during policy configuration in step 2.

Specified user: Select this option if you want to grant operation permissions to specified sub-accounts, root accounts, or cloud services. During policy configuration in step 2, you need to further specify the account UINs.

Resource Scope

The whole bucket: if you want to grant bucket configuration permissions or set the resource scope to the entire bucket, you can select this option to automatically add the entire bucket as a resource for you during policy configuration in step 2.

Specified directory: select this option if you want to restrict the resource scope to a specified folder. During policy configuration in step 2, you need to further specify the specific directory.

Template

A template is a collection of operations that you want to authorize. COS provides you with different recommended templates depending on the combination of authorized users and resource scope you choose. If the templates provided by COS do not meet your requirements, you can skip this step and add or delete authorized operations during policy configuration in step 2.

Custom (no preset configuration): If you do not need to use a template, select this option and add policies as needed during policy configuration in step 2.

Other templates: COS provides you with different recommended templates depending on the combination of authorized users and resource scope you choose. After you select a template, COS automatically adds the corresponding operation permissions for you during policy configuration in step 2.

Templates are described in the following table.

Grantee	Resource Scope	Policy Template	Description
All combinations		Custom	For any combination of authorized users and resource scopes, this template does not provide any preset policies. You can add policies during policy configuration in step 2.
All users (allow anonymous access)	The whole bucket	Read-Only objects (listing objects is not included)	For anonymous users, COS provides you with recommended templates for reading files (such as downloading files) and writing files (such as uploading and modifying files).COS's recommended templates do not list all objects in your bucket, and sensitive permissions, such as read and write permissions and bucket configuration permissions, are not allowed to improve data security. You can add or delete operation permissions during policy configuration in step 2 as needed.
		Read/Write objects (listing	

		objects is not included)	
	Specified directory	Read-Only objects (listing objects is not included)	
		Read/Write objects (listing objects is not included)	
Specified user	The whole bucket	Read-Only objects (listing objects is not included)	<p>COS provides the most recommended templates for the combination of Specified user and The whole bucket</p> <p>. In addition to reading, writing, and listing files, COS provides the following sensitive permission templates for trusted users:</p> <p>Read/Write buckets and object ACLs: get and modify buckets and object ACLs. Options include GetObjectACL, PutObjectACL, GetBucketACL, and PutBucketACL.</p> <p>General bucket configuration items: non-sensitive permissions such as bucket tagging, CORS, and origin-pull.</p> <p>Bucket sensitive configuration item: sensitive permissions such as bucket policies, bucket ACLs, and bucket deletion. Sensitive permissions should be used with caution.</p>
		Read-Only objects (listing objects is included)	
		Read/Write objects (listing objects is not included)	
		Read/Write objects (listing objects is included)	
		Read/Write buckets and object ACLs	
		General bucket configuration items	

		Bucket sensitive configuration item	
	Specified directory	Read-Only objects (listing objects is not included)	For the combination of Specified user and Specified directory, COS provides you with recommended templates for reading files (such as downloading files) and writing files (such as uploading and modifying files), as well as recommended templates for listing objects. If you need to grant read, write, and list permissions to a specified folder to a specified user, this combination is recommended. You can add or delete operation permissions during policy configuration in step 2 as needed.
		Read-Only objects (listing objects is included)	
		Read/Write objects (listing objects is not included)	
		Read/Write objects (listing objects is included)	

Step 2: configure the policy

Based on the combination of authorized users, specified directories, and templates you select in step 1, COS automatically adds operations, authorized users, and resources to the configuration policy for you. If you specify a user and a directory, you need to specify the user UIN and directory during policy configuration.

Note:

To authorize the permissions of a directory, you need to add `/*` to the resource path entered. For example, to authorize the `test` directory, you need to enter `test/*`.

If the recommended templates provided by COS do not meet your requirements, you can add or delete authorized users, resources, and operations in this step.

The configuration items are described as follows:

Effect: select **Allow** or **Deny**, corresponding to `allow` or `deny` in the policy syntax.

User: add or delete authorized users. Options include **Everyone** (`*`), **Root account**, **Sub-account**, and **Cloud service**.

Resource: add the whole bucket or a specific directory resource.

Operation: add or delete authorized operations as needed.

Condition: you can specify conditions for permission authorization. For example, you can specify a user access IP.

JSON

If you are familiar with bucket policies, you can click the target bucket, choose **Permission Management** > **Permission Policy Settings** > **JSON**, and write the bucket policy in JSON language.

After writing the bucket policy, you can add it via [API](#) or [SDK](#).

JSON policy example

The following policy is to allow root account 1000000000001 (APPID: 12500000000) to grant sub-account 1000000000011 with all operation permissions of the objects in the `folder/sub-folder` directory in the `examplebucket-bj` bucket of the Beijing region.



```
{
  "Statement": [
    {
      "Principal": {
        "qcs": [
          "qcs::cam::uin/1000000000001:uin/1000000000011"
        ]
      },
      "Effect": "Allow",
      "Action": [
        "name/cos:*"
      ]
    }
  ]
}
```

```
    ],  
    "Resource": [  
        "qcs::cos:ap-beijing:uid/1250000000:examplebucket-bj-1250000000/folder/sub-  
    ]  
}  
],  
"version": "2.0"  
}
```

Operation Methods

COS allows you to add bucket policies by using the COS console, APIs, or SDKs. The COS console provides the Visual Editor configuration mode and common authorization templates to help users that are unfamiliar with the policy language to quickly add policies.

Operation Method		Description
Console		Web page, intuitive and easy to use
API		RESTful API, sending requests directly to COS
SDK	JavaScript	Rich SDK demos, supporting various programming languages
	Node.js	
	Mini Program	

More Bucket Policy Examples

[Granting authorization through bucket policies](#)

[Granting Sub-Account Under One Root Account Permission to Manipulate Buckets Under Another Root Account](#)

Note:

When adding an access policy, be sure to grant the minimum permissions needed to satisfy your business needs. There may be data security risks if you grant other users the permission to access all of your resources

(resource:*) or all operations (action:*) .

The following are examples of bucket policies used to limit subnets, principals and VPC IDs.

Example 1

Limit the IP range in the subnet to 10.1.1.0/24 and the VPC ID to aqp5jrc1:



```
{
  "Statement": [
    {
      "Action": [
        "name/cos:*"
      ],
      "Condition": {
        "ip_equal": {
          "qcs:ip": [
```

```
        "10.1.1.0/24"
      ]
    },
    "string_equal": {
      "vpc:requester_vpc": [
        "vpc-aqp5jrc1"
      ]
    }
  },
  "Effect": "deny",
  "Principal": {
    "qcs": [
      "qcs::cam::anyone:anyone"
    ]
  },
  "Resource": [
    "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
  ]
},
"version": "2.0"
}
```

Example 2

Limit the VPC ID to aqp5jrc1 and specify the principal and bucket:



```
{
  "Statement": [
    {
      "Action": [
        "name/cos:*"
      ],
      "Condition": {
        "string_equal": {
          "vpc:requester_vpc": [
            "vpc-aqp5jrc1"
          ]
        }
      }
    }
  ]
}
```

```
    }
  },
  "Effect": "allow",
  "Principal": {
    "qcs": [
      "qcs::cam::uin/1000000000001:uin/1000000000002"
    ]
  },
  "Resource": [
    "qcs::cos:ap-beijing:uid/1250000000:examplebucket-1250000000/*"
  ]
},
"version": "2.0"
}
```

User Policy

Last updated : 2024-03-25 15:51:41

In the [CAM console](#), you can use your Tencent Cloud root account to create CAM users and grant them with Tencent Cloud resource permissions by associating them with policies.

Overview

In [CAM](#), you can grant different permissions to different types of users under your root account. These permissions are described in the access policy language and are granted by user, so they are called **user policies**.

Differences between a user policy and a bucket policy

The biggest difference between a user policy and a bucket policy is that the user policy only describes effect, action, resource, and condition (optional), but not principal. Therefore, for a user policy:

You need to write a user policy first, and then associate it manually with a sub-user, a user group or a role.

A user policy **cannot grant anonymous users access to resources or operations**.

Preset policy and custom policy

User policies are classified into [preset policies and custom policies](#). You can [associate a preset policy for authorization](#) or [write a user policy](#) and associate it for authorization. For more information, see CAM's [Authorization Guide](#).

Application Scenarios

If you want to specify what operations a user can perform, you are advised to configure user policy. You can search for a CAM user and check the permissions of the user's user groups to see what operations the user can perform. A user policy is recommended in scenarios where you want to:

Configure COS service-level permissions such as the permissions for bucket creation (PutBucket) and bucket listing (GetService).

Grant permissions on all COS buckets and objects under your root account.

Grant the same permissions to a large number of CAM users under the root account.

User Policy Syntax

Policy syntax

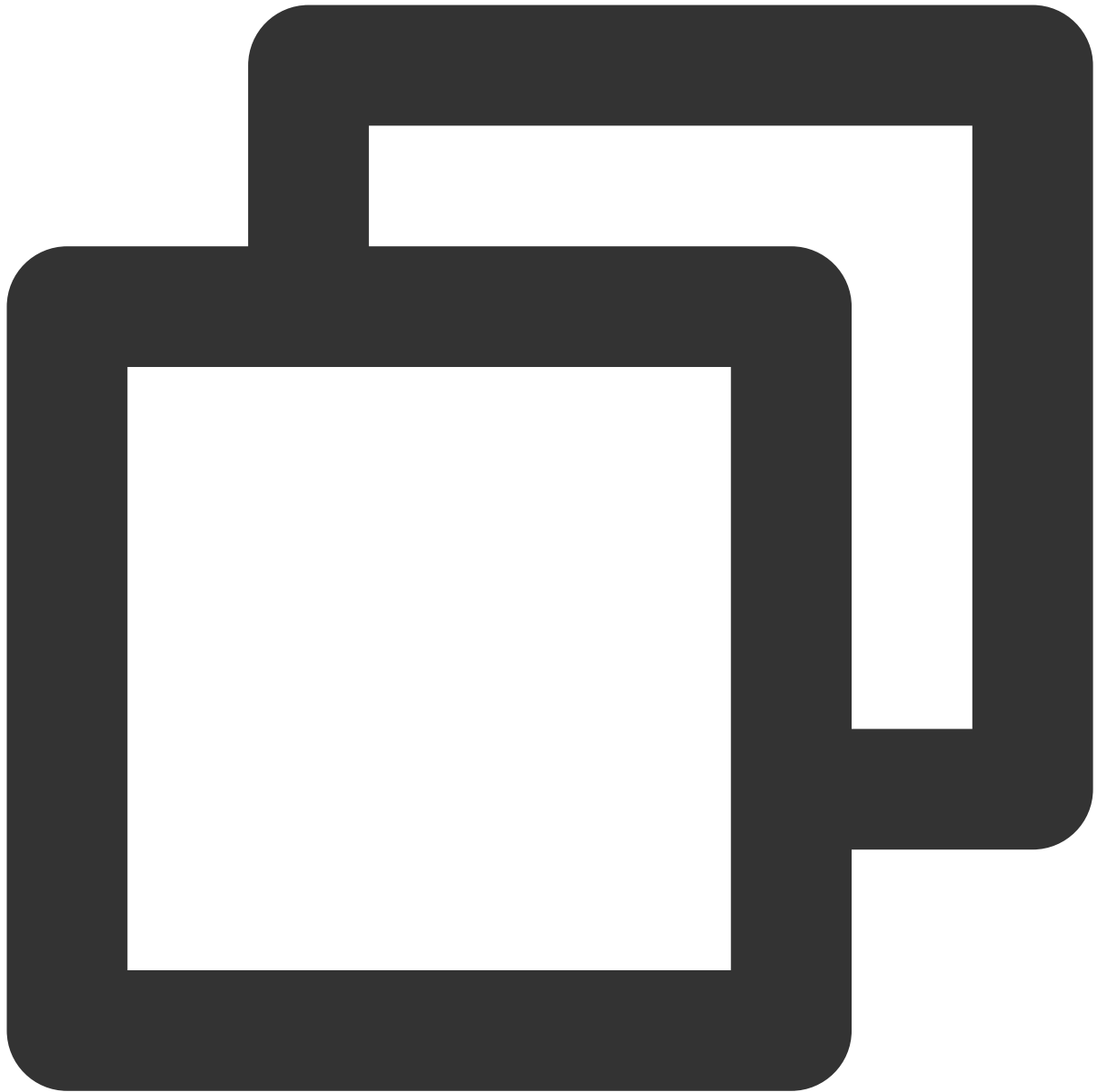
Same as a bucket policy, a user policy is described in JSON language and its syntax complies with the unified specifications of the [access policy language](#). The access policy language contains the following basic elements: principal, effect, action, resource, and condition. However, a user policy is directly associated with a user or user group, and therefore you do not need to specify the principal element in a user policy.

The following table compares a user policy and a bucket policy:

Element	User Policy	Bucket Policy
Principal	No input	Required
Effect	Required	Required
Action	Required	Required
Resource	Required	Resources in the current bucket
Condition	Optional	Optional

Policy examples

The following typical user policy example grants the permission to perform all COS operations on the bucket `examplebucket-1250000000` in Guangzhou. You need to save the policy and then associate it with a [CAM](#) sub-user, a user group or a role before it takes effect.



```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["cos:*"],
      "Resource": [
        "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*",
        "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/"
      ]
    }
  ]
},
```

```
"Version": "2.0"
}
```

Granting Sub-account COS Access Permission with User Policy

Prerequisites

You have already created a CAM sub-account. If you haven't, see [Create a sub-account](#) to create one.

Directions

CAM provides [preset policies and custom policies](#). A preset policy is a policy preset in the system provided by CAM. For COS related preset policies, see [Preset Policy](#). A custom policy allows users to customize elements such as resources and actions. The following describes how to create a custom policy to grant permissions to a sub-account:

1. Log in to the [CAM console](#).
2. Choose **Policy > Create Custom Policy > Create by Policy Syntax** to go to the policy creation page.
3. You can select **Blank Template** to customize a permission policy as needed or select a COS-associated **system template**. The following uses **Blank Template** as an example.
4. Select **Blank Template** and enter your policy syntax. The policy syntax must contain the following elements:

resource: resource to authorize access to

All resources (`"*"`)

Specified bucket (`"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"`)

Specified directory or object in a bucket (`"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/test/*"`)

action: action to authorize

effect: select `"Allow"` or `"Deny"`

condition: optional

COS provides user policy examples. You can copy and paste the policy content in the following documents into the **Policy Content** box and click **Done**.

[Granting Sub-accounts Access to COS](#)

[Working with COS API Access Policies](#)

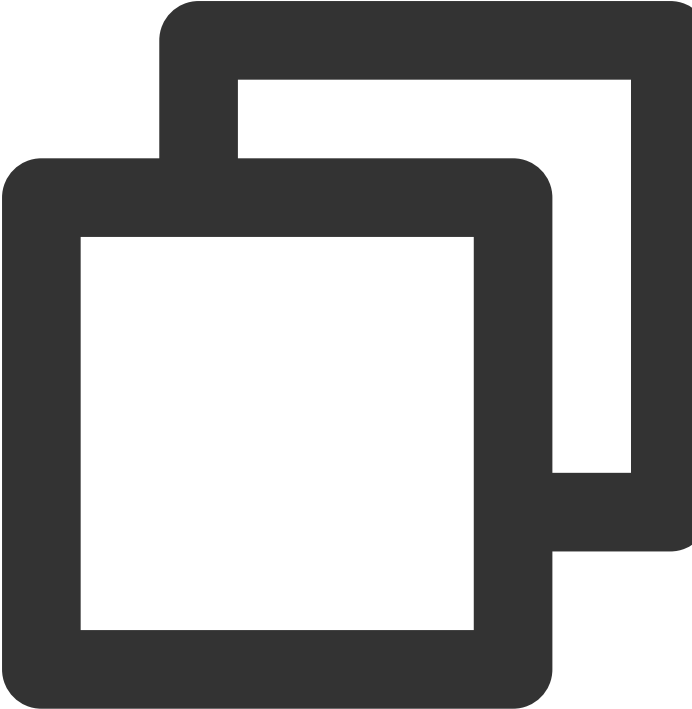
5. After creating the policy, you can go to **Policy > Custom Policy** in the [CAM console](#) to view the created custom policy and associate it with sub-accounts.
6. Select the sub-account and click **Confirm** to complete the authorization.

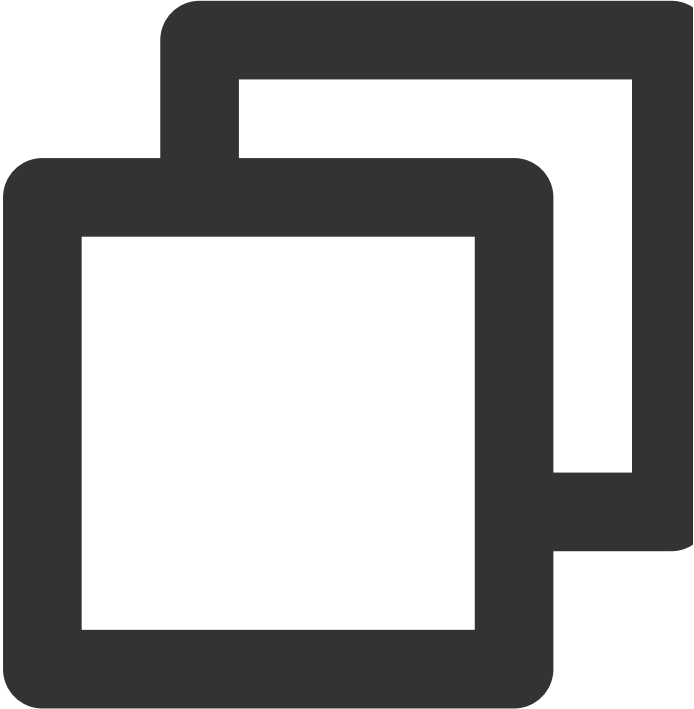
Preset Policy

1. CAM provides some preset policies. You can view them (filtered by "COS") in **Policy > Preset Policy** in the [CAM console](#).
2. Click a policy name and choose **Policy Syntax > JSON** to view the policy context. In a preset policy, `resource` is set to `"*"` (indicating all resources in COS) and its value cannot be modified. If you want to grant permissions on certain COS buckets or objects, you can copy the JSON preset policy to create a [Custom Policy](#).

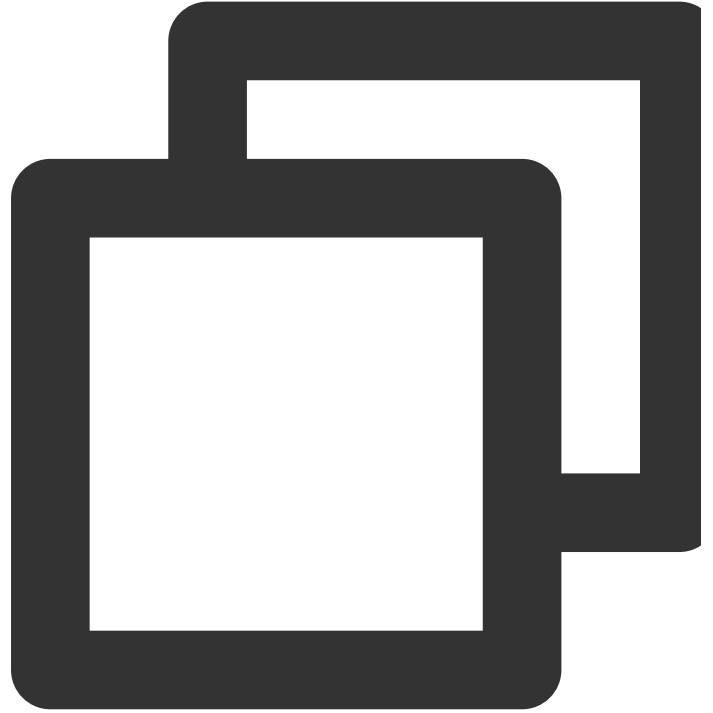
Table 1 and Table 2 list the COS related preset policies provided by CAM and related descriptions.

Table 1. COS Preset Policies

Preset Policy	Description	JSON Policy
QcloudCOSBucketConfigRead	Permission to read COS bucket configuration	 <pre>{ "version": "2.0", "statement": [{ "effect": "allow", "action": ["cos:GetBucket*", "cos:HeadBucket"], "resource": "*" }] }</pre>

		<pre>} </pre>
QcloudCOSBucketConfigWrite	Permission to modify COS bucket configuration	 <pre>{ "version": "2.0", "statement": [{ "effect": "allow", "action": ["cos:PutBucket*"], "resource": "*" }] }</pre>
QcloudCOSDataFullControl	Permission to read, write, delete,	

and list data
in the COS
bucket (all
access
permissions)



```
{
  "version": "2.0",
  "statement": [
    {
      "effect": "allow",
      "action": [
        "cos:GetService",
        "cos:GetBucket",
        "cos:ListMultipartUploads",
        "cos:GetObject*",
        "cos:HeadObject",
        "cos:GetBucketObjectVersion",
        "cos:OptionsObject",
        "cos:ListParts",
        "cos:DeleteObject",
        "cos:PostObject",
        "cos:PostObjectRestore",
        "cos:PutObject*",
        "cos:InitiateMultipartUploa",
        "cos:UploadPart",
        "cos:UploadPartCopy",
        "cos:CompleteMultipartUploa",
        "cos:AbortMultipartUpload",
        "cos:DeleteMultipleObjects"
      ]
    }
  ]
}
```

		<pre> "cos:AppendObject"], "resource": "*" }] }</pre>
--	--	-----------------------------------------------------------------------------------

Table 2. Relationships between COS actions and preset policies

Action Purpose	Action	QcloudCOS Bucket ConfigRead	QcloudCOS Bucket ConfigWrite	QcloudCOS Data FullControl	QcloudCOS Data ReadOnly
List buckets	GetService	✗	✗	✓	✗
Create a bucket	PutBucket	✗	✓	✗	✗
Delete a bucket	DeleteBucket	✗	✗	✗	✗
Get basic bucket information	HeadBucket	✓	✗	✗	✗
Get bucket configuration items	GetBucket*	✓	✗	✗	✗
Modify bucket configuration items	GetBucket*	✗	✓	✗	✗
Get bucket access permissions	GetBucketAcl	✓	✗	✗	✗
Modify bucket access permissions	PutBucketAcl	✗	✓	✗	✗

List objects in a bucket	GetBucket	✓	✗	✓	✗
List all objects in a bucket and their historical version information	GetBucketObjectVersions	✓	✗	✓	✗
Upload an object	PutObject	✗	✗	✓	✗
Multipart upload	ListParts InitiateMultipartUpload UploadPart UploadPartCopy CompleteMultipartUpload AbortMultipartUpload ListMultipartUploads	✗	✗	✓	✗
Append an object	AppendObject	✗	✗	✓	✗
Download an object	GetObject	✗	✗	✓	✓
View object metadata	HeadObject	✗	✗	✓	✓
Issue a preflight request for CORS	OptionsObject	✗	✗	✓	✓

More User Policy Examples

[Granting Sub-accounts Access to COS](#)

[Working with COS API Access Policies](#)

ACL

Last updated : 2024-03-25 15:33:39

Concepts

An ACL is described in the XML language. It is a list of specified grantees and permissions granted, which is associated with resources. Each bucket and object has an associated ACL to grant basic read and write permissions to anonymous users or other Tencent Cloud root accounts.

Note:

Here are some limits on the use of ACLs associated with resources:

The resource owner always has the FULL_CONTROL permission on the resource, which cannot be revoked or modified.

An anonymous user cannot be the resource owner. In this case, the object owner is the bucket owner (Tencent Cloud root account).

Permissions can only be granted to Tencent Cloud CAM root accounts or preset user groups, but not custom user groups. Besides, it's not recommended to grant permissions to any sub-users.

No conditions should be imposed on the permissions.

"Deny" permission is not supported.

A resource can have up to 100 ACL policies.

Use Cases

Note:

Enabling anonymous access (Public Read) is highly risky due to hotlinking risks. In cases where Public Read is required, you can [set hotlink protection](#) to improve security.

Use ACLs if you only need to set some simple access permissions for buckets and objects or enable anonymous access. However, in most cases, you are advised to use bucket policies or user policies, which are more flexible.

ACLs are recommended in scenarios where you want to:

Set simple access permissions.

Set access permissions quickly in the console.

Make an object, directory, or bucket accessible to all anonymous internet users.

ACL Elements

Identity (Grantee)

A supported grantee can be a CAM root account or a preset CAM user group.

Note:

When you grant access permissions to another Tencent Cloud root account, this root account can grant access permissions to its sub-users, user groups or roles.

In COS, it is strongly recommended not to grant the WRITE, WRITE_ACP or FULL_CONTROL permission to any anonymous user or CAM user group. Otherwise, the user group can upload, download or delete your resources, which will bring security risks to your account, such as data loss and fee deduction.

Grantees supported in bucket or object ACLs include:

Cross-account: use the root account ID to get the account ID, e.g., `1000000000001`, in [Account Info](#).

Preset user group: tag a preset user group using a URI tag. The following user groups are supported:

Anonymous user group - `http://cam.qcloud.com/groups/global/AllUsers`, which indicates that anyone can access resources without authorization, regardless of whether the request is signed or unsigned.

Authenticated user group - `http://cam.qcloud.com/groups/global/AuthenticatedUsers`, which indicates that users who have registered a Tencent Cloud CAM account can access resources.

Action permissions

Tencent Cloud COS supports a range of action sets on resource ACLs, which include different actions on bucket ACLs and object ACLs respectively.

Actions on buckets

Here is a list of actions supported in bucket ACLs:

Action Set	Description	Allowed Actions
READ	Lists objects	HeadBucket, GetBucketObjectVersions, ListMultipartUploads
WRITE	Uploads, overwrites and deletes objects	PutObject, PutObjectCopy, PostObject, InitiateMultipartUpload, UploadPart, UploadPartCopy, CompleteMultipartUpload, DeleteObject
READ_ACP	Reads bucket ACLs	GetBucketAcl
WRITE_ACP	Writes to bucket ACLs	PutBucketAcl
FULL_CONTROL	A collection of the above sets	A collection of the above actions

Note:

Please proceed with caution when you grant the WRITE, WRITE_ACP or FULL_CONTROL permission on buckets. Granting the WRITE permission will allow the grantee to overwrite or delete any existing object.

Actions on objects

Here is a list of actions supported in object ACLs:

Action Set	Description	Allowed Actions
READ	Reads objects	GetObject, GetObjectVersion, HeadObject
READ_ACP	Reads object ACLs	GetObjectAcl, GetObjectVersionAcl
WRITE_ACP	Writes to object ACLs	PutObjectAcl, PutObjectVersionAcl
FULL_CONTROL	A collection of the above sets	A collection of the above action sets

Note:

The WRITE action set is not supported for objects.

Preset ACL

Tencent Cloud COS supports a group of preset ACLs for authorization, making it much easier to describe simple permissions. When using a preset ACL for description, you need to put the `x-cos-acl` header in the PUT Bucket/Object or PUT Bucket/Object ACL and describe the required permission. If an XML description is also carried in the request body, the description in the header is used and the XML description in the request body is ignored.

Preset ACLs for buckets

Name	Description
private	The creator (root account) has the FULL_CONTROL permission, while others have no permissions. (Default)
public-read	The creator has the FULL_CONTROL permission, and the anonymous user group has the READ permission.
public-read-write	Both the creator and the anonymous user group have the FULL_CONTROL permission, which is generally not recommended.
authenticated-read	The creator has the FULL_CONTROL permission, and the authenticated user group has the READ permission.

Preset ACLs for objects

Name	Description
default	An empty description. The explicit settings of each level of directory and the settings of the bucket are used to determine whether a request is allowed. (Default)
private	The creator (root account) has the FULL_CONTROL permission, while others have no

	permissions.
public-read	The creator has the FULL_CONTROL permission, and the anonymous user group has the READ permission.
authenticated-read	The creator has the FULL_CONTROL permission, and the authenticated user group has the READ permission.
bucket-owner-read	The creator has the FULL_CONTROL permission, and the bucket owner has the READ permission.
bucket-owner-full-control	Both the creator and the bucket owner have the FULL_CONTROL permission.

Note:

The public-read-write permission is not supported for objects.

Example

Bucket ACL

When you create a bucket, COS creates a default ACL to grant the resource owner the full control over the resource (FULL_CONTROL), as shown in the following example:



```
<AccessControlPolicy>
  <Owner>
    <ID>Owner-Cononical-CAM-User-Id</ID>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee>
        <ID>Owner-Cononical-CAM-User-Id</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
```

```
</AccessControlList>  
</AccessControlPolicy>
```

Object ACL

When you create an object, COS does not create an ACL by default, and the object owner is the bucket owner. The object's permissions inherited from its bucket are the same with the access permissions of the bucket. Since the object does not have a default ACL, it will follow the definition of visitors and behaviors in the bucket policy to determine if the request is granted. For more information, see [Access Policy Language Overview](#).

If you need to grant additional access permissions to the object, you can add more ACLs to describe the access permissions of the object. For example, grant an anonymous user the permission to read a single object, as shown below:



```
<AccessControlPolicy>
  <Owner>
    <ID>Owner-Cononical-CAM-User-Id</ID>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee>
        <ID>Owner-Cononical-CAM-User-Id</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
```

```
<Grant>
  <Grantee>
    <URI>http://cam.qcloud.com/groups/global/AllUsers</URI>
  </Grantee>
  <Permission>READ</Permission>
</Grant>
</AccessControlList>
</AccessControlPolicy>
```

Tag-based project resource management

Last updated : 2024-03-25 15:33:39

Overview

Note:

This document describes how to use tags and tag authentication to manage project resources in the tag system. It is applicable to scenarios where a user used projects in the legacy console and granted sub-accounts access through project authentication.

Project management is the process of centrally managing resources based on projects. You can add Tencent Cloud product resources that support project management to a project. Then, select **Policies > Create Custom Policy > Create by Product Feature or Project Permission** in the [CAM console](#) to generate a project policy. You can associate the project policy with project-related users or user groups, so that they can get the permission to manipulate project resources.

The legacy COS console enables permission management operations based on projects. However, a project policy grants the full access to all resources under all products included in the project. It **cannot meet the needs for multi-dimensional tagging and categorization or manage permissions in a refined manner**. In the new COS console, COS only supports tag-based permission management for project resources.

COS uses the tag service for compatibility with the legacy project feature. In the tag service system, a project is a special tag with the tag key being `project`. You can still create a project and then create a bucket under it in the project console. COS will automatically double-write the project association of a bucket during bucket creation to the tag service, so that the association can be displayed in the console.

Note:

If you need to manage buckets in a categorized manner, we recommend you directly use tags instead of projects to implement tasks such as permission control and cost allocation. For more information on how to add tags in the console, see [Setting Bucket Tags](#).

For more information on projects, see [Cloud Access Management](#). For more information on the tag service, see [Tag](#). For more information on benefits of tags, see [Cloud Access Management](#).

Granting a Sub-Account Access to a Project

You can grant a sub-account access to a project by following the steps below:

1. Log in to the [Project console](#), create a project, name it, and submit it. Then, create a resource such as a bucket or CVM instance under it.

If you already have a project with storage or computing resources, you can skip this step.

2. After you create a project and bind a resource to it, enter the [Policy Management](#) page, click **Create Custom Policy > Authorize by Tag**, select COS and the operation permission to be granted, select the corresponding project tag, and grant the sub-account access to all resources under it.

3. Click **Next**, grant the permission to the selected user/user group/role, and click **Complete**.

Note:

The default policy is to grant the sub-account access to all resources under the project tag. If you want the sub-account to perform only specified operations on certain resources under the tag, you can change the `action` (for specifying operations) and `resource` (for specifying resources) in the policy syntax as instructed in [Syntax Structure](#).

If you want the sub-account to be able to create buckets, you also need to grant it the `PUT Bucket` permission. On the [Policy Management](#) page, click **Create Custom Policy > Create by Policy Generator** and grant the sub-account the corresponding permission.

Authorization Methods

Last updated : 2024-03-25 15:33:39

Differences Between Bucket Policies, User Policies, Bucket ACLs, and Object ACLs

The differences between bucket policies, user policies, bucket ACLs, and object ACLs are as follows, which are further illustrated in Table 1:

User policies grant permissions based on users. Bucket policies, bucket ACLs, and object ACLs grant permissions based on resources.

User policies and bucket policies are based on the access policy language and therefore provide higher permission control flexibility. They allow you to grant the permission of each specific action and the permission effect can be set to `deny` or `allow`. Bucket ACLs and object ACLs are based on ACLs and therefore provide relatively lower permission control flexibility. They allow you to grant only basic read and write permissions.

User policies cannot be used to grant permissions to anonymous users. Bucket policies, bucket ACLs, and object ACLs can be used to grant permissions to anonymous users.

User policies are configured in the CAM console. Bucket policies, bucket ACLs, and object ACLs are configured in the COS console.

Table 1. Differences between authorization methods

Item		User policy	Bucket policy	Bucket ACL	Object ACL
Classification		User-based authorization	Resource-based authorization	Resource-based authorization	Resource-based authorization
Permission control flexibility		High flexibility	High flexibility	Low flexibility	Low flexibility
Console configuration		CAM console	COS console	COS console	COS console
Access control element	User	All CAM identities (sub-accounts, roles, etc.) management by the current account	Sub-accounts, root accounts, anonymous users	Sub-accounts, other root accounts, anonymous users	Sub-accounts, other root accounts, anonymous users
			Sub-account Tencent Cloud services, roles, etc.		
		Before cross-account	Directly supports cross-account authorization.		

		authorization, you need to add the target account as your collaborator.			
	Effect	Allow + Deny	Allow + Deny	Allow only	Allow only
	Resource	All cloud resources, all COS buckets, specified buckets (prefixes, objects, etc.)	Specified buckets (prefixes, objects, etc.)	Entire bucket	Specified objects
	Action	Every specific action	Every specific action (excluding bucket creation and bucket listing)	Simplified read and write permissions	Simplified read and write permissions
	Condition	Supported	Supported	Not supported	Not supported

How to Choose an Appropriate Authorization Method?

The differences listed in Table 1 show the advantages and disadvantages of different authorization methods. You can choose an appropriate authorization method based on your own needs.

But whichever authorization method you choose, you are advised to maintain consistency as much as possible.

Otherwise, with the increase of bucket policies, user policies, and ACLs, it will become increasingly difficult to maintain permissions.

When do I use an ACL?

Note:

Enabling anonymous access (Public Read) is highly risky due to hotlinking risks. In cases where Public Read is required, you can [set hotlink protection](#) to improve security.

Use ACLs if you only need to set some simple access permissions for buckets and objects or enable anonymous access. However, in most cases, you are advised to use bucket policies or user policies, which are more flexible.

ACLs are recommended in scenarios where you want to:

- Set simple access permissions.

- Set access permissions quickly in the console.

- Make an object, directory, or bucket accessible to all anonymous internet users.

Each account supports only 1,000 ACLs. If this upper limit is exceeded, select bucket policies or user policies instead.

When do I use a user policy?

If you want to specify what operations a user can perform, you are advised to configure user policy. You can search for a CAM user and check the permissions of the user's user groups to see what operations the user can perform. A user policy is recommended in scenarios where you want to:

Configure COS service-level permissions such as the permissions for bucket creation and bucket listing.

Grant permissions on all COS buckets and objects under your root account.

Grant the same permissions to a large number of CAM users under the root account.

When do I use a bucket policy?

If you want to specify which users can access a COS bucket, you are advised to configure a bucket policy. You can search for a bucket and check the bucket's policy to see who can access the bucket. A bucket policy is recommended in scenarios where you want to:

Authorize a specific bucket.

Authorize a specific action with the authorization effect set to `deny` or `allow`, which is not supported by ACL.

Use cross-account authorization and anonymous user authorization, which are not supported by user policies.

Access Policy Language Overview

Last updated : 2024-03-25 15:33:39

Note:

When adding access policies to a sub-user or collaborator, be sure to grant the minimum set of API access permissions needed to satisfy your business needs. There may be data security risks associated with granting excessive access to all of your resources `(resource:*)` or all operations `(action:*)` .

Overview

An access policy that employs the JSON-based access policy language is used to grant access to Cloud Object Storage (COS) resources. You can authorize a specified principal to perform actions on a specified COS resource through the access policy language.

The language describes the basic elements and usage of a bucket policy. For more information, see [CAM Policy Management](#).

Elements in an Access Policy

The access policy language contains the following basic elements:

principal: describes the entity to be authorized by the policy. This includes users (root accounts, sub-accounts, anonymous users) and user groups. This element can only be used in bucket access policies, rather than user access policies.

statement: describes the detailed information of one or more permissions. This element includes a permission or a permission set of multiple elements such as effect, action, resource, and condition. A policy has only one statement.

effect: describes the result of a statement. The result can be "allow" or an explicit "deny". This element is required.

action: describes the allowed or denied operation. An operation can be an API (described using the prefix "name") or a feature set (a set of specific APIs prefixed with "permid"). This element is required.

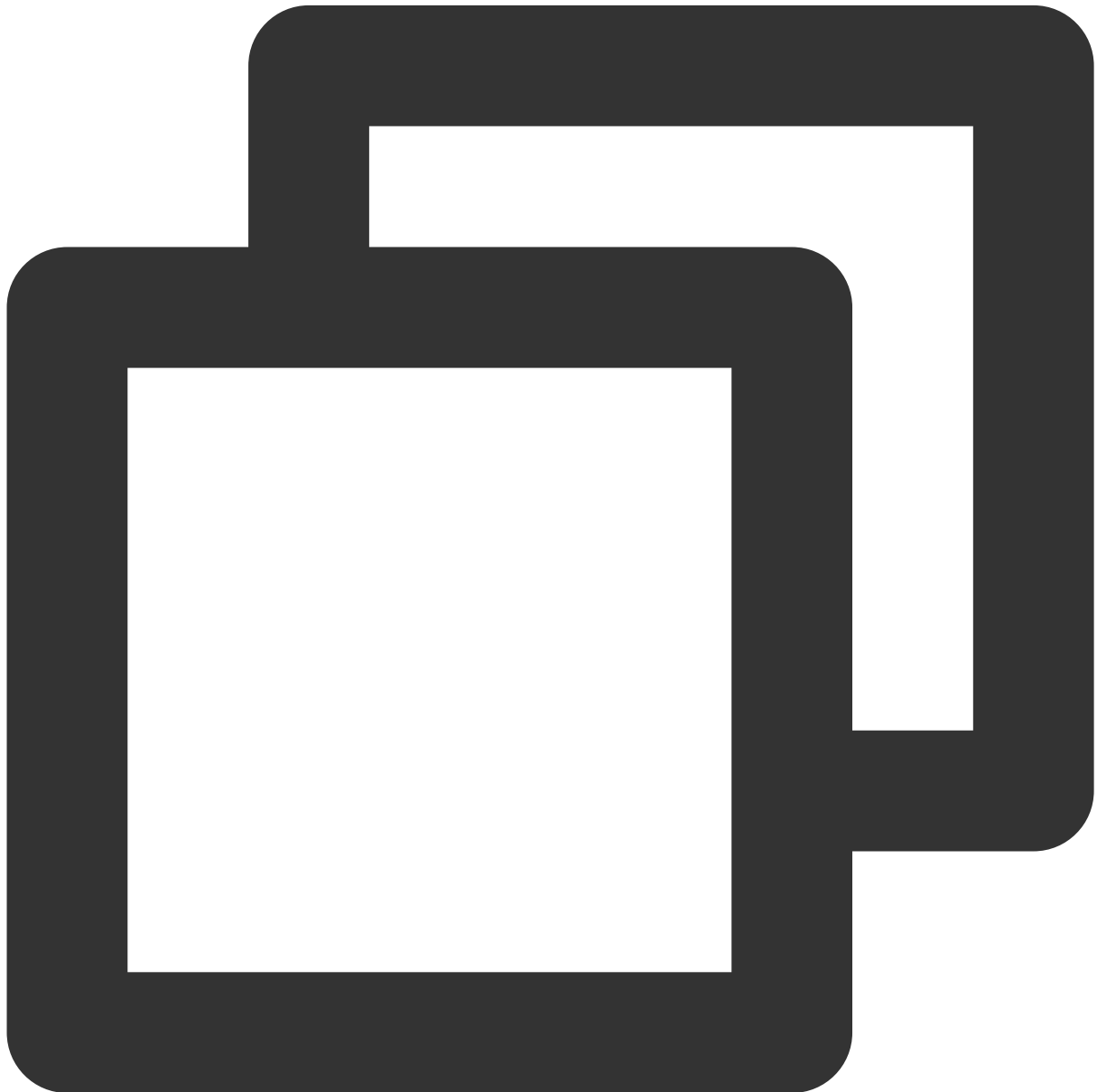
resource: describes the resource to which the permission applies. A resource is described in a six-segment format. Detailed resource definitions vary by product. For more information on how to specify a resource, see the documentation for the product whose resources you are writing a statement for. This element is required.

condition: describes the condition for the policy to take effect. A condition consists of operator, action key, and action value. A condition value may contain information such as time and IP address. Some services allow you to specify additional values in a condition. This element is optional.

How to Use Elements

Specifying a principal

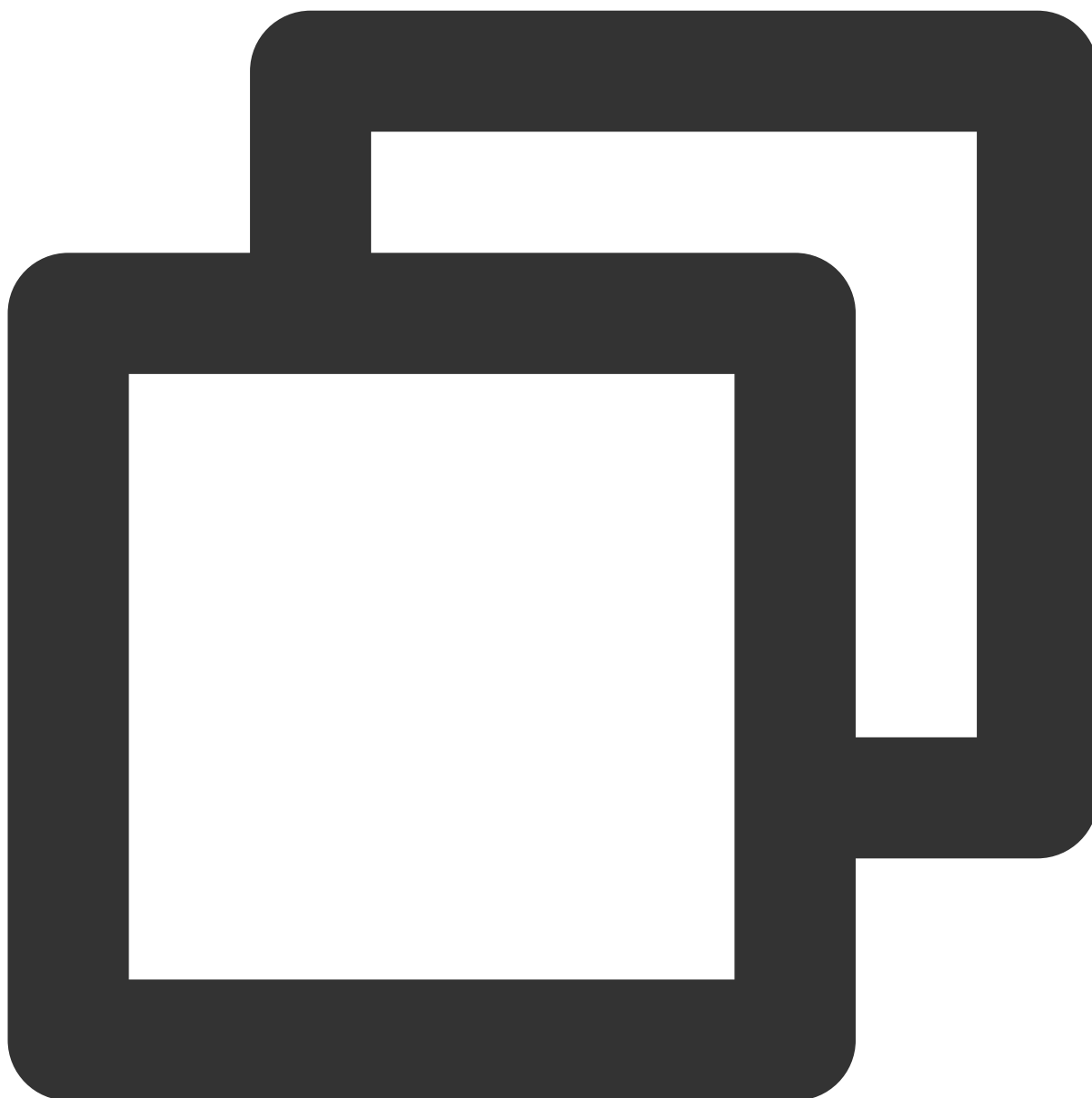
A principal is used to specify the user, account, service, or another entity that is allowed or not allowed to access resources. It only works in buckets, and is not required in user policies because these policies are directly added to the specific user. The following example specifies a principal.



```
"principal":{  
  "qcs":[  
    "qcs::cam::uin/1000000000001:uin/1000000000001"  
  ]  
}
```

```
]
}
```

Grant permissions to an anonymous user:



```
"principal":{
  "qcs":[
    "qcs::cam::anonymous:anonymous"
  ]
}
```

Grant permissions to a root account (UIN: 100000000001):



```
"principal":{  
  "qcs":[  
    "qcs::cam::uin/1000000000001:uin/1000000000001"  
  ]  
}
```

Grant permissions to a sub-account (UIN: 100000000011) under the root account (UIN: 100000000001):

Note:

Before performing the operation, make sure that the sub-account has been added to the sub-account list of the root account.

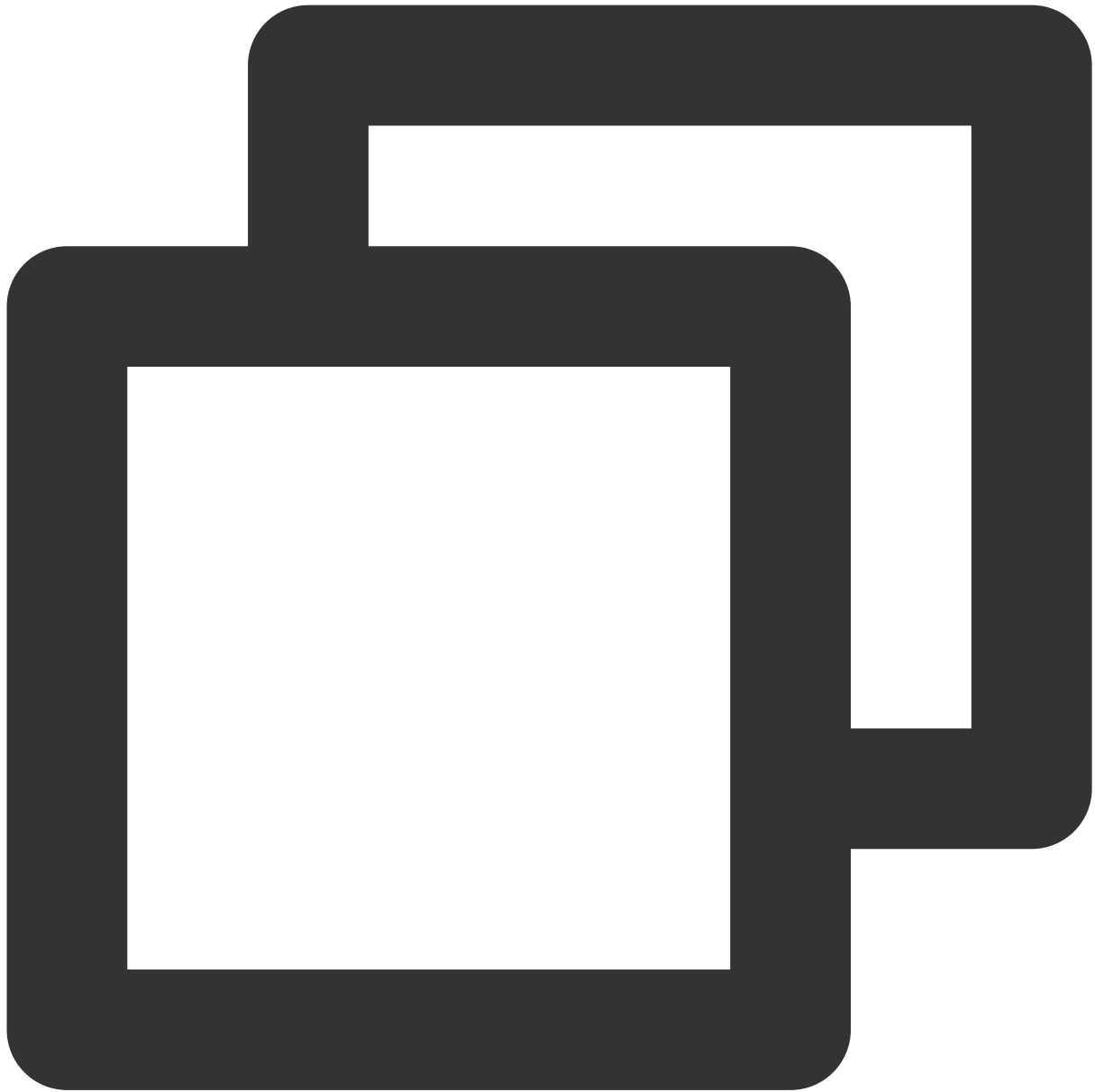


```
"principal":{  
  "qcs":[  
    "qcs::cam::uin/1000000000001:uin/1000000000011"  
  ]  
}
```

Specifying an effect

If you don't explicitly grant access to (`allow`) a resource, access is implicitly denied (`deny`). You can also deny access to a resource, which you might do to make sure that a user cannot access it, even if a different policy grants

access. The following example specifies an `allow` effect.



```
"effect" : "allow"
```

Specifying an action

COS defines actions that you can specify in a policy. The actions are exactly the same as COS API operations. The following lists only part of the actions on buckets and objects. For more information, see [Operation List](#).

Actions on buckets

--	--

Description	API
name/cos:GetService	GET Service
name/cos:GetBucket	GET Bucket (List Objects)
name/cos:PutBucket	PUT Bucket
name/cos>DeleteBucket	DELETE Bucket

Actions on objects

Description	API
name/cos:GetObject	GET Object
name/cos:PutObject	PUT Object
name/cos:HeadObject	HEAD Object
name/cos>DeleteObject	DELETE Object

The following example specifies an action that is allowed:



```
"action": [
  "name/cos:GetObject",
  "name/cos:HeadObject"
]
```

Specifying a resource

The `resource` element describes one or multiple operation objects including COS buckets or objects. All the resources can be described using the following 6-segment format.



```
qcs:project_id:service_type:region:account:resource
```

The parameters are described as follows:

Parameter	Description	Required
qcs	Abbreviation of the qcloud service, which refers to Tencent Cloud services.	Yes
project_id	Describes the project information, which is only used to enable compatibility with legacy CAM logic.	No

service_type	Describes the abbreviation of the product such as COS.	Yes
region	Describes the region information. For more information, see Regions & Endpoints supported by Tencent Cloud COS.	Yes
account	Describes the root account information of the resource owner. Currently, <code>uin</code> or <code>uid</code> can be used to describe the resource owner. <code>uin</code> is the UIN of the root account. It is expressed in the format of <code>uin/\${OwnerUin}</code> , for example, <code>uin/1000000000001</code> . <code>uid</code> is the app ID of the root account. It is expressed in the format of <code>uid/\${appid}</code> , for example, <code>uid/1250000000</code> . Currently, COS resource owners are all described using <code>uid</code> , i.e., the app ID of the root account.	Yes
resource	Describes the detailed resource information. In COS, a resource is described using the bucket XML API access domain name.	Yes

The following example specifies the bucket `examplebucket-1250000000` .



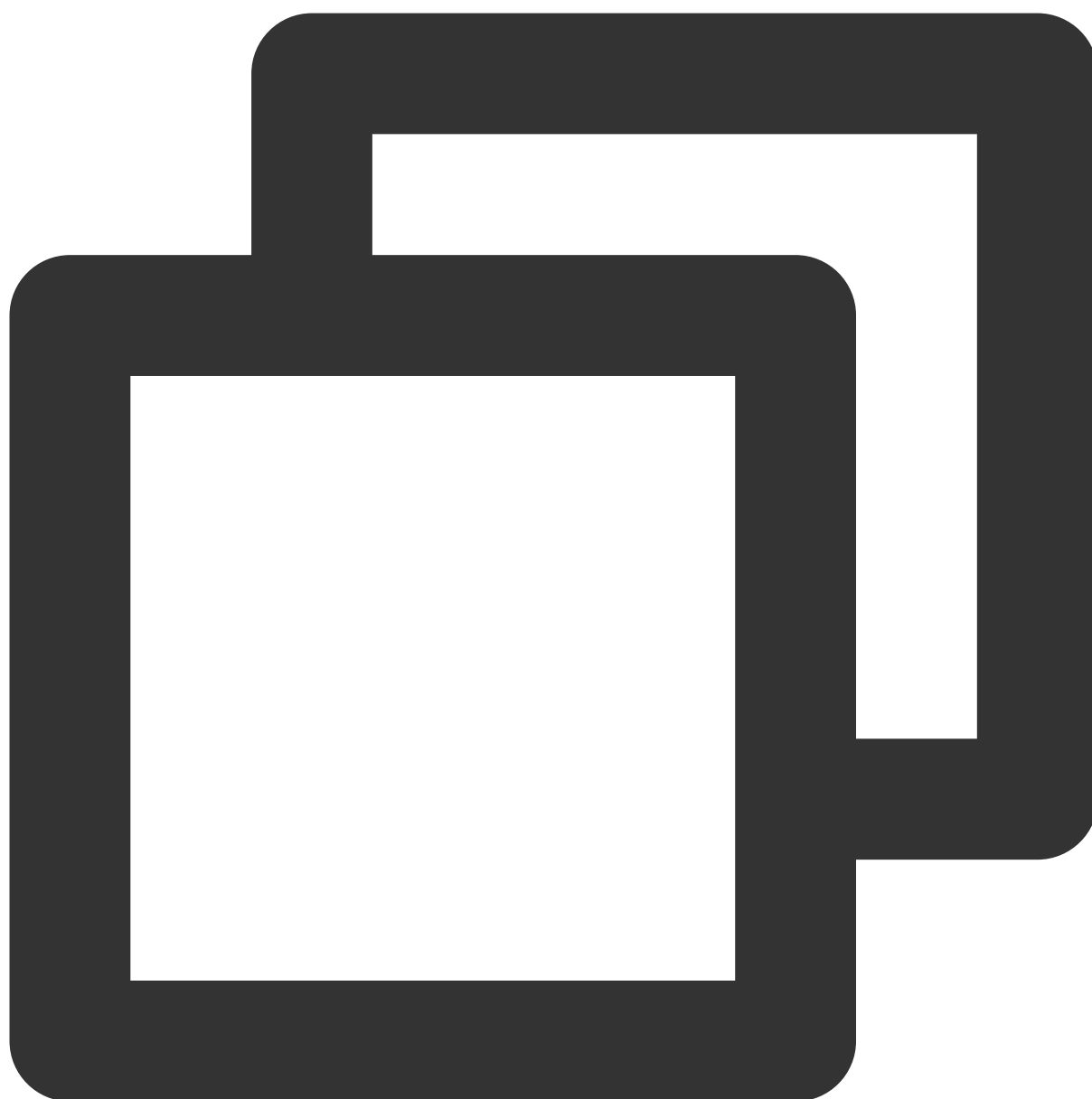
```
"resource": ["qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"]
```

The following example specifies all objects in the `/folder/` folder in the bucket `examplebucket-1250000000` .



```
"resource": ["qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/folder/
```

The following example specifies the `/folder/exampleobject` object in the bucket `examplebucket-1250000000` .



```
"resource": ["qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/folder/
```

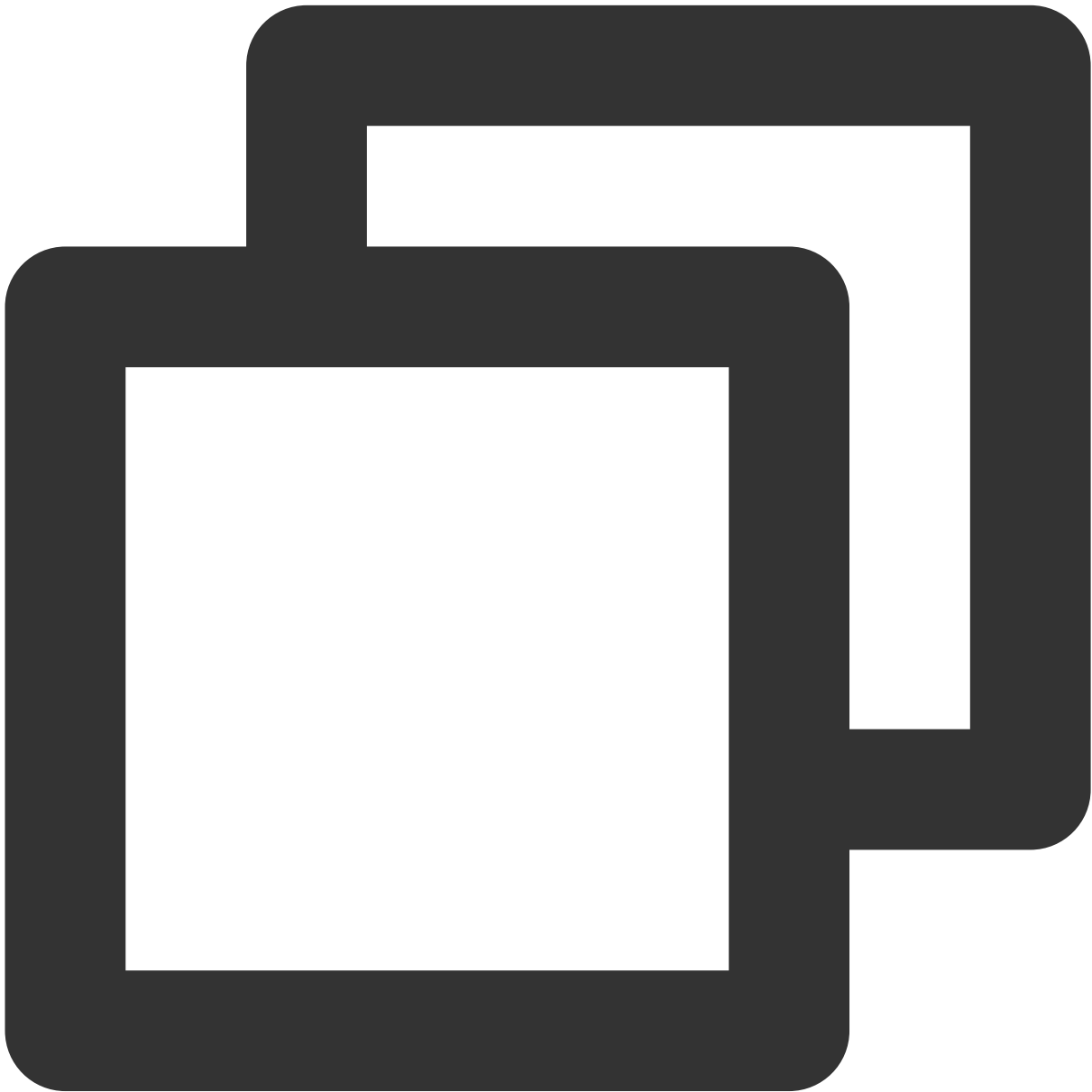
Specifying a condition

The access policy language allows you to specify conditions when granting permissions, for example, to limit the access source of a user. Here is a list of supported conditional operators as well as general condition keys and examples.

Conditional Operator	Description	Condition Name	Example
----------------------	-------------	----------------	---------

ip_equal	IP is equal to	qcs:ip	{"ip_equal":{"qcs:ip ":"10.121.2.0/24"}}
ip_not_equal	IP is not equal to	qcs:ip	{"ip_not_equal":{"qcs:ip":["10.121.1.0/24","10.121.2.0/24"]}}

The following example allows access to only IPs in the 10.121.2.0/24 IP range.



```
"ip_equal":{"qcs:ip ":"10.121.2.0/24"}
```

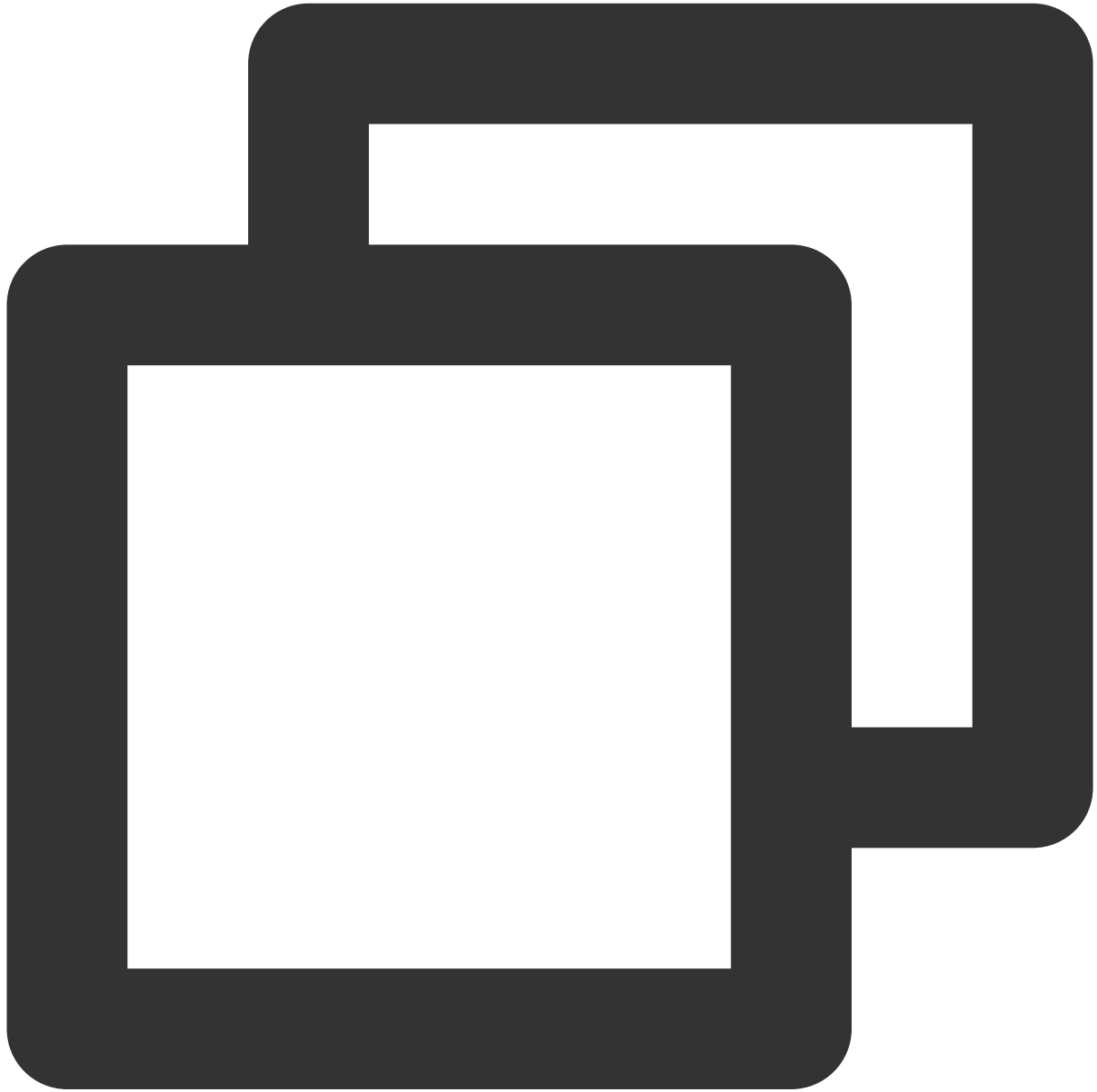
The following example allows access to only the IPs 101.226.100.185 and 101.226.100.186.



```
"ip_equal":{
  "qcs: ip": [
    "101.226.100.185",
    "101.226.100.186"
  ]
}
```

Examples

If, when the access source IPs are 101.226.100.185 and 101.226.100.186, the root account allows anonymous users to perform GET (download) and HEAD actions on the objects in the bucket `examplebucket-1250000000` in South China, no authentication is required. For more information, see [Cases of Permission Setting](#).



```
{
  "version": "2.0",
  "principal": {
    "qcs": [
      "qcs: : cam: : anonymous: anonymous"
    ]
  },
}
```

```
"statement": [  
  {  
    "action": [  
      "name/cos: GetObject",  
      "name/cos: HeadObject"  
    ],  
    "condition": {  
      "ip_equal": {  
        "qcs: ip": [  
          "101.226.100.185",  
          "101.226.100.186"  
        ]  
      }  
    },  
    "effect": "allow",  
    "resource": [  
      "qcs: : cos: ap-guangzhou: uid/1250000000: examplebucket-1250000000"  
    ]  
  }  
]  
}
```

Conditions

Last updated : 2024-03-25 15:33:39

Overview

Conditions are part of the access policy language. A complete condition includes the following elements:

Condition key: Specifies the type of the condition, for example, user access source IP and authorization time.

Condition operator: Specifies the condition determination method.

Condition value: Specifies the value of the condition key.

For more information, see [Conditions](#) of Access Management.

Note:

When using condition keys to write a bucket policy, follow the principle of least privilege, add the corresponding condition keys only to applicable requests (actions), and avoid using the * wildcard when specifying the actions. Using the wildcard will cause the requests to fail.

When you create a policy in the CAM console, pay attention to the syntax format. The syntax elements of `version` , `principal` , `statement` , `effect` , `action` , `resource` , and `condition` must all begin with a letter in the same letter case.

Condition Example

In the following bucket policy example, the condition specifies that the `cos:PutObject` authorization operation can be completed only in the 10.217.182.3/24 or 111.21.33.72/24 IP range. In the condition:

The **condition key** is `qcs:ip` , indicating that the condition type is IP.

The **condition operator** is `ip_equal` , indicating that the condition determination method is to determine whether IP addresses match.

The **condition value** is the `["10.217.182.3/24", "111.21.33.72/24"]` array, listing the specified values for condition determination. If the user's IP is in any of the specified IP ranges in the array, the condition is determined as true.



```
{
  "version": "2.0",
  "statement": [
    {
      "principal": {
        "qcs": [
          "qcs::cam::uin/1250000000:uin/1250000001"
        ]
      },
      "effect": "allow",
      "action": [
```

```

        "name/cos:PutObject"
    ],
    "resource": [
        "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
    ],
    "condition": {
        "ip_equal": {
            "qcs:ip": [
                "10.217.182.3/24",
                "111.21.33.72/24"
            ]
        }
    }
}
]
}

```

Condition Keys Supported by COS

Note:

Currently, the `tls-version` condition key is supported only in the Beijing region. It will be supported in other regions later.

COS supports two types of condition keys: condition keys applicable to all requests, including IP, VPC, and HTTPS, and condition keys from request headers and request parameters and generally applicable only to requests that need to carry the request headers or request parameters. For the descriptions and use cases of these condition keys, see [Descriptions and Use Cases of Condition Keys](#).

Note:

Conditions and condition keys are applicable only to the access management of user requests. When the lifecycle and bucket replication rules are valid, actions such as deletion and replication are initiated by COS rather than users and therefore are not in the applicable scope of condition keys.

Condition keys applicable to all requests

Condition keys applicable to all requests are `qcs:ip`, `qcs:vpc`, and `cos:secure-transport`, which indicate the source IP range of the request, source VPC ID of the request, and whether the request uses HTTPS, respectively.

Condition Key	Applicable Request	Meaning	Type
<code>cos:secure-transport</code>	All requests	Whether the request uses HTTPS	Boolean
<code>qcs:ip</code>	All requests	Source IP range of the request	IP

<code>qcs:vpc</code>	All requests	Source VPC ID of the request	String
<code>cos:tls-version</code>	All HTTPS requests	TLS version used by HTTPS requests	Numeric

Condition keys from request headers and request parameters

Because different requests have different request headers (`Header`) and request parameters (`Param`), condition keys from request headers and request parameters are applicable only to requests that contain such request headers or request parameters.

For example, the condition key `cos:content-type` is applicable to upload requests (such as `PutObject`) that need to use the request header `Content-Type` , while the condition key `cos:response-content-type` is applicable only to `GetObject` requests because only `GetObject` requests support the request parameter `response-content-type` .

The table below lists the condition keys from request headers and request parameters and the corresponding applicable requests.

Condition Key	Applicable Request	Check Request Header or Request Parameter	Type
<code>cos:x-cos-storage-class</code>	<code>PutObject</code> <code>PostObject</code> <code>InitiateMultipartUpload</code> <code>AppendObject</code>	Request header: <code>x-cos-storage-class</code>	String
<code>cos:versionid</code>	<code>GetObject</code> <code>DeleteObject</code> <code>PostObjectRestore</code> <code>PutObjectTagging</code> <code>GetObjectTagging</code> <code>DeleteObjectTagging</code> <code>HeadObject</code>	Request parameter: <code>versionid</code>	String
<code>cos:prefix</code>	<code>GetBucket (List Objects)</code> <code>GET Bucket Object versions</code> <code>List Multipart Uploads</code> <code>ListLiveChannels</code>	Request parameter: <code>prefix</code>	String
<code>cos:x-cos-acl</code>	<code>PutObject</code> <code>PostObject</code> <code>PutObjectACL</code> <code>PutBucket</code> <code>PutBucketACL</code> <code>AppendObject</code> <code>Initiate Multipart Upload</code>	Request header: <code>x-cos-acl</code>	String

<code>cos:content-length</code>	This request header has a wide applicable scope, typically requests with request bodies.	Request header: <code>Content-Length</code>	Numeric
<code>cos:content-type</code>	This request header has a wide applicable scope, typically requests with request bodies.	Request header: <code>Content-Type</code>	String
<code>cos:response-content-type</code>	GetObject	Request parameter: <code>response-content-type</code>	String
<code>qcs:request_tag</code>	PutBucket PutBucketTagging	Request header: x-cos-tagging Request parameter: tagging	String

Condition Operators

COS supports the following condition operators, which are applicable to condition keys of the String, Numeric, Boolean, and IP types.

Condition Operator	Description	Type
<code>string_equal</code>	String equal to (case-sensitive)	String
<code>string_not_equal</code>	String not equal to (case-sensitive)	String
<code>string_like</code>	String similar to (case-sensitive). Currently, wildcards (*) can be prefixed or suffixed to the string, for example, <code>image/*</code> .	String
<code>ip_equal</code>	IP equal to	IP
<code>ip_not_equal</code>	IP not equal to	IP
<code>numeric_equal</code>	Number equal to	Numeric
<code>numeric_not_equal</code>	Number not equal to	Numeric
<code>numeric_greater_than</code>	Number greater than	Numeric
<code>numeric_greater_than_equal</code>	Number greater than or equal to	Numeric
<code>numeric_less_than</code>	Number less than	Numeric
<code>numeric_less_than_equal</code>	Number less than or equal to	Numeric

`_if_exist`

You can add `_if_exist` to the end of all the preceding condition operators to form new condition operators, such as `string_equal_if_exist`. The differences between condition operators with and without `_if_exist` are as follows:

For a condition operator without `_if_exist`, such as `string_equal`, it is considered that the condition is met (`False`) by default if the request does not contain the specified request header or parameter.

For a condition operator with `_if_exist`, such as `string_equal_if_exist`, it is considered that the condition is met (`True`) by default if the request does not contain the specified request header or parameter.

License request example

Example 1: allow downloading objects of a specified version

In the bucket policy in this example, `Effect` is `allow`, allowing `GetObject` requests where the request parameter `versionid` is `MTg0NDUxNTc1NjIzMTQ1MDAwODg`. According to the `allow` authorization policy, if the condition is met (`True`), the request will be allowed; if the condition is not met (`False`), the request will not be allowed and will fail.



```
{
  "version": "2.0",
  "statement": [
    {
      "principal": {
        "qcs": [
          "qcs::cam::uin/1250000000:uin/1250000001"
        ]
      },
      "effect": "allow",
      "action": [
```

```

        "name/cos:GetObject"
    ],
    "condition":{
        "string_equal":{
            "cos:versionid":"MTg0NDUxNTc1NjIzMTQ1MDAwODg"
        }
    },
    "resource":[
        "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
    ]
}
]
}

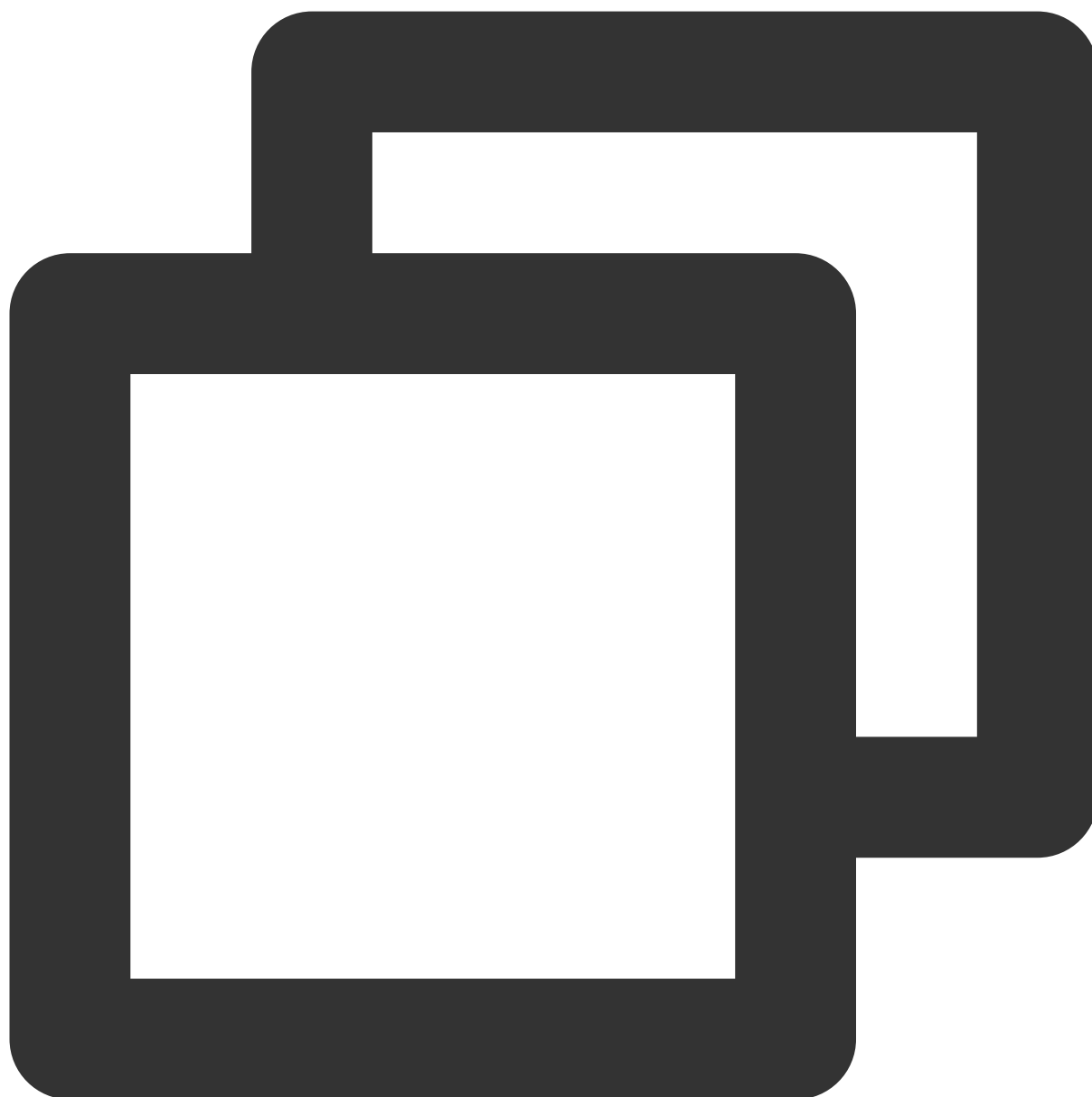
```

The table below lists the condition meeting and request allowing details of the condition operators `string_equal` and `string_equal_if_exist`.

Condition Operator	Request	Condition Met	Request Allowed
<code>string_equal</code>	Without <code>versionid</code>	FALSE	No
<code>string_equal_if_exist</code>	Without <code>versionid</code>	TRUE	Yes
<code>string_equal</code>	With <code>versionid</code> , whose value is specified	TRUE	Yes
<code>string_equal_if_exist</code>	With <code>versionid</code> , whose value is specified	TRUE	Yes
<code>string_equal</code>	With <code>versionid</code> , whose value is not specified	FALSE	No
<code>string_equal_if_exist</code>	With <code>versionid</code> , whose value is not specified	FALSE	No

Example 2: disallow downloading objects of a specified version

In the bucket policy in this example, `Effect` is `deny`, disallowing `GetObject` requests where the request parameter `versionid` is `MTg0NDUxNTc1NjIzMTQ1MDAwODg`. According to the `deny` authorization policy, if the condition is met (`True`), the request will fail; if the condition is not met (`False`), the request will not be denied.



```
{
  "version": "2.0",
  "statement": [
    {
      "principal": {
        "qcs": [
          "qcs::cam::uin/1250000000:uin/1250000001"
        ]
      },
      "effect": "deny",
      "action": [
```

```

        "name/cos:GetObject"
    ],
    "condition":{
        "string_equal":{
            "cos:versionid":"MTg0NDUxNTc1NjIzMTQ1MDAwODg"
        }
    },
    "resource":[
        "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
    ]
}
]
}

```

The table below lists the condition meeting and request denial details of the condition operators `string_equal` and `string_equal_if_exist`.

Condition Operator	Request	Condition Met	Request Denied
<code>string_equal</code>	Without <code>versionid</code>	FALSE	No
<code>string_equal_if_exist</code>	Without <code>versionid</code>	TRUE	Yes
<code>string_equal</code>	With <code>versionid</code> , whose value is specified	TRUE	Yes
<code>string_equal_if_exist</code>	With <code>versionid</code> , whose value is specified	TRUE	Yes
<code>string_equal</code>	With <code>versionid</code> , whose value is not specified	FALSE	No
<code>string_equal_if_exist</code>	With <code>versionid</code> , whose value is not specified	FALSE	No

Notes

Special characters require URL encoding

Special characters in request parameters all require URL encoding, and therefore bucket policies that use condition keys from the request parameters require URL encoding as well. For example, if you are to use the

`cos:response-content-type` condition key in a bucket policy, the condition value `image/jpeg` must be encoded (URL encoding) into `image%2Fjpeg` before it is entered into the bucket policy.

Principle of least privilege and no * wildcard

When using condition keys, comply with the principle of least privilege, add only actions for which you want to set permissions, and avoid using the * wildcard. Misuse of the * wildcard may cause some requests to fail. In the example below, requests other than `GetObject` do not support using the request parameter `response-content-type`.

For "deny + string_equal_if_exist", if the condition key does not exist in the request, it is considered `True` by default. Therefore, when you initiate requests such as `PutObject` and `PutBucket`, the `deny` statement will be met and the requests will be denied.

For "allow + string_equal", if the condition key does not exist in the request, it is considered `False` by default. Therefore, when you initiate requests such as `PutObject` and `PutBucket`, the `allow` statement will not be met and the requests will not be allowed.



```
{
  "version": "2.0",
  "statement": [
    {
      "principal": {
        "qcs": [
          "qcs::cam::uin/1250000000:uin/1250000001"
        ]
      },
      "effect": "allow",
      "action": [
```

```

        "*"
    ],
    "resource": [
        "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
    ],
    "condition": {
        "string_equal": {
            "cos:response-content-type": "image%2Fjpeg"
        }
    }
},
{
    "principal": {
        "qcs": [
            "qcs::cam::uin/1250000000:uin/1250000001"
        ]
    },
    "effect": "deny",
    "action": [
        "*"
    ],
    "resource": [
        "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
    ],
    "condition": {
        "string_not_equal_if_exist": {
            "cos:response-content-type": "image%2Fjpeg"
        }
    }
}
]
}

```

Alternatively, you can use "allow + string_equal_if_exist" and "deny + string_not_equal" to allow requests without the `response-content-type` request parameter.

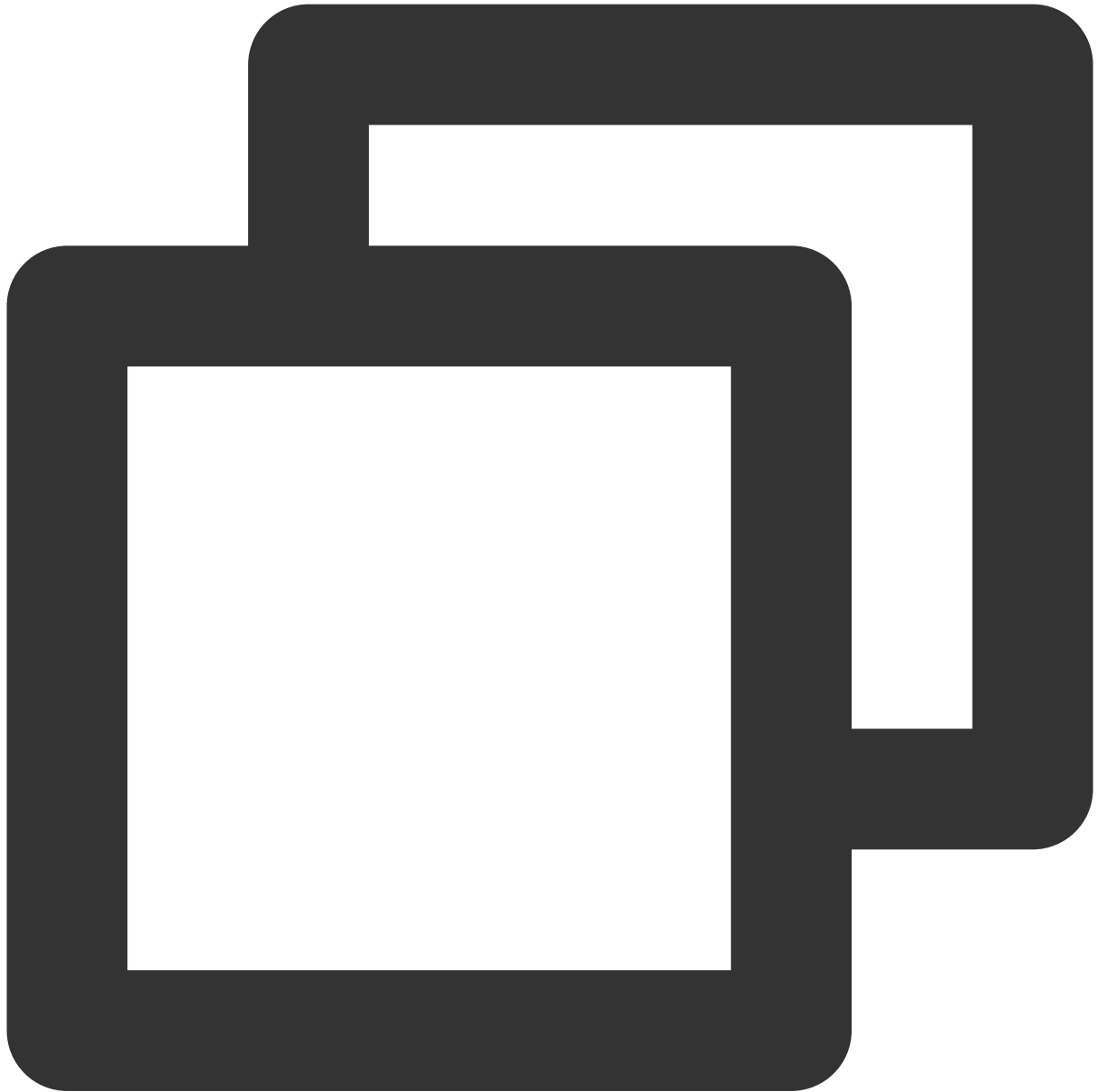
For "deny + string_equal", if the condition key does not exist in the request, it is considered `False` by default.

Therefore, when you initiate requests such as `PutObject` and `PutBucket`, the `deny` statement will not be met and the requests will not be denied.

For "allow + string_equal_if_exist", if the condition key does not exist in the request, it is considered `True` by default. Therefore, when you initiate requests such as `PutObject` and `PutBucket`, the `allow` statement will be met and the requests will be allowed.

However, if you use condition operators in this way, you will not be able to restrict whether a `GetObject` request carries the `response-content-type` request parameter. A `GetObject` request without the `response-content-type` request parameter will be allowed by default like other requests. Only when the `GetObject`

request carries the `response-content-type` request parameter, you can use your specified condition to check whether the content of the request parameter is the same as what you expect to implement conditional authorization.



```
{
  "version": "2.0",
  "statement": [
    {
      "principal": {
        "qcs": [
          "qcs::cam::uin/1250000000:uin/1250000001"
        ]
      }
    }
  ]
}
```

```

    },
    "effect": "allow",
    "action": [
        "*"
    ],
    "resource": [
        "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
    ],
    "condition": {
        "string_equal_if_exist": {
            "cos:response-content-type": "image%2Fjpeg"
        }
    }
},
{
    "principal": {
        "qcs": [
            "qcs::cam::uin/1250000000:uin/1250000001"
        ]
    },
    "effect": "deny",
    "action": [
        "*"
    ],
    "resource": [
        "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
    ],
    "condition": {
        "string_not_equal": {
            "cos:response-content-type": "image%2Fjpeg"
        }
    }
}
]
}

```

Therefore, the more secure way is to comply with the principle of least privilege and restrict the action to

`GetObject` requests without using the `*` wildcard.

As shown in the example policy below, the policy condition strictly specifies that authorization is performed only for

`GetObject` requests carrying the `response-content-type` request parameter with value

`image%2Fjpeg`.

Other requests are not affected by the policy in this example, and can be authorized separately based on the principle of least privilege.



```
{
  "version": "2.0",
  "statement": [
    {
      "principal": {
        "qcs": [
          "qcs::cam::uin/1250000000:uin/1250000001"
        ]
      },
      "effect": "allow",
      "action": [
```

```
        "name/cos:GetObject"
    ],
    "resource": [
        "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
    ],
    "condition": {
        "string_equal": {
            "cos:response-content-type": "image%2Fjpeg"
        }
    }
},
{
    "principal": {
        "qcs": [
            "qcs::cam::uin/1250000000:uin/1250000001"
        ]
    },
    "effect": "deny",
    "action": [
        "name/cos:GetObject"
    ],
    "resource": [
        "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
    ],
    "condition": {
        "string_not_equal_if_exist": {
            "cos:response-content-type": "image%2Fjpeg"
        }
    }
}
]
```

Request Methods

Accessing COS Using a Permanent Key

Last updated : 2024-03-25 15:33:39

Background

With RESTful APIs, you can initiate anonymous HTTP requests or signed HTTP requests to COS. Anonymous requests are typically used for scenarios that require public access, such as hosting static websites; and signed requests are required in most other scenarios.

Compared with an anonymous request, a signed request carries an additional signature value. The signature is an encrypted string generated based on the key (SecretId/SeretKey) and request information. The SDK automatically calculates the signature. You only need to set the key when initializing user information and do not need to worry about signature calculation. For requests initiated through RESTful APIs, COS needs to calculate signatures based on the signature algorithm and add them to the requests.

Getting a Permanent Key

You can log in to the CAM console and go to the [Manage API Key](#) page to get a permanent key. A permanent key consists of a SecretId and a SecretKey. It represents the permanent identity of your account and does not expire.

SecretId: used to identify the API caller.

SecretKey: used to encrypt the signature string and server-side authentication signature string.

Accessing COS Using a Permanent Key

Accessing COS using an API request

When using an API request, you must use a signed request for a private bucket. A signature is generated based on a permanent key and put into the `Authorization` header to form a signed request. When the request is sent to COS, COS verifies whether the signature matches the request.

Note:

Because the signature generation algorithm is complex, you are advised to use an SDK to initiate a request and skip this step.

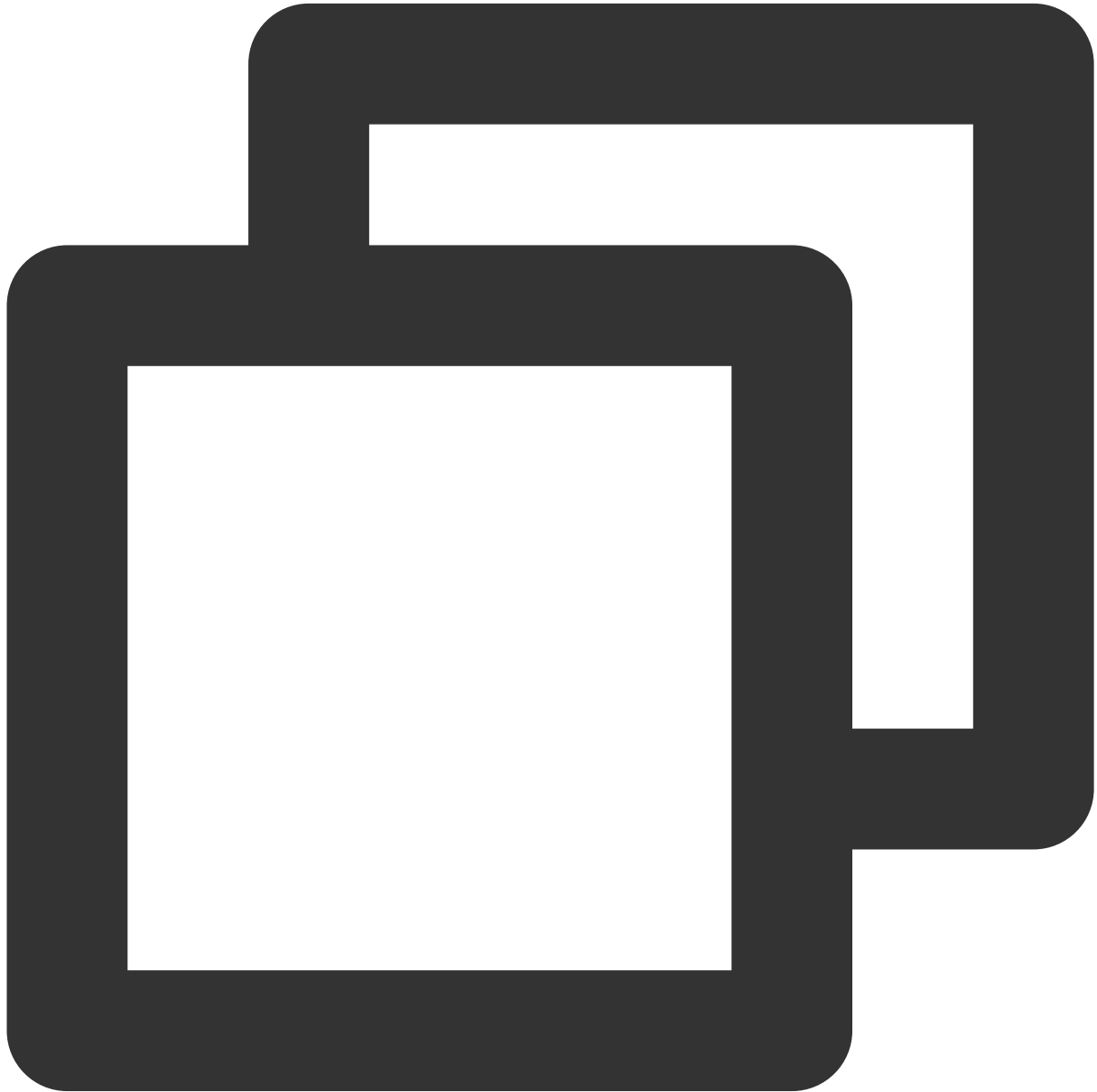
1. Use a permanent key to generate a signature.

For the signature algorithm, see [Request Signature](#). COS provides a signature generation tool. You can also use a

COS SDK to generate signatures. For more information, see [Implementing Signature in SDK](#). You can also write a program to generate signatures. However, this method is not recommended because the signature algorithm is complex.

2. Enter the signature into the `Authorization` header.

When initiating an API request, enter the signature into the standard HTTP `Authorization` header. The following is an example of a `GetObject` request:



```
GET /<ObjectKey> HTTP/1.1
Host: <BucketName-APPID>.cos.<Region>.myqcloud.com
Date: GMT Date
```

```
Authorization: q-sign-algorithm=sha1&q-ak=SecretId&q-sign-time=KeyTime&q-key-time=K
```

Accessing COS using an SDK tool

1. Initialize the identity information with the permanent key.

After installing the SDK tool, enter the permanent key (SecretId and SecretKey) of the root account or sub-account to initialize the user identity information.

2. Directly use the SDK tool to initiate requests to COS.

After initialization, you can directly use the SDK tool for upload, download, and other basic operations, without the need to generate signatures like using API requests, because the SDK tool generates signatures based on keys on your behalf and initiates requests to COS.

The following is an example of the corresponding Java SDK code. For demos of other languages, see the corresponding quick start documentation in [SDK Overview](#).



```
// 1. Initialize the user credentials (secretId, secretKey).  
// Log in to the [CAM console](https://console.tencentcloud.com/cam/capi) to view a  
String secretId = "SECRETID";  
String secretKey = "SECRETKEY";  
COSCredentials cred = new BasicCOSCredentials(secretId, secretKey);  
// 2. Set the bucket region. For abbreviations of COS regions, please visit https://  
// `clientConfig` contains the set methods to set region, HTTPS (HTTP by default),  
Region region = new Region("COS_REGION");  
ClientConfig clientConfig = new ClientConfig(region);  
// The HTTPS protocol is recommended.  
// Starting from 5.6.54, HTTPS is used by default.
```



```
clientConfig.setHttpProtocol(HttpProtocol.https);  
// 3. Generate a COS client.  
COSClient cosClient = new COSClient(cred, clientConfig);
```

Accessing COS Using a Temporary Key

Last updated : 2024-03-25 15:33:39

Temporary Key Overview

Temporary keys are temporary access credentials provided by Security Token Service (STS). A temporary key consists of three parts: TmpSecretId, TmpSecretKey, and token. Compared with a permanent key, a temporary key has the following characteristics:

A temporary key has a short validity period (30 minutes to 36 hours), and will not expose a permanent key, reducing the risk of account disclosure.

When getting a temporary key, you can set temporary permissions by passing in the `policy` parameter to further constrain the user's permission scope.

Therefore, temporary keys are suitable for temporary authorization scenarios such as frontend direct upload.

Compared with permanent keys, temporary keys are more secure to be distributed to untrusted users.

Getting a Temporary Key

You can get a temporary key via COS STS SDK or by calling the STS API directly.

For more information, see [Generating and Using Temporary Keys](#).

Permissions of Temporary Keys

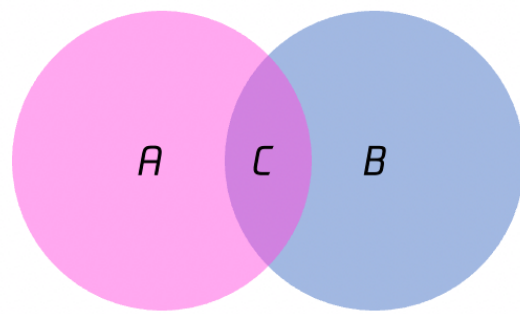
You need a CAM user account (Tencent Cloud root account or sub-account) to apply for a temporary key. When getting a temporary key, you can pass in the `policy` parameter to add a temporary policy to the temporary key to further constrain the user's permission scope.

If the `policy` parameter is not set, the temporary key obtained has the same permissions as those of the CAM user.

If the `policy` parameter is set, the temporary key obtained will, based on the CAM user's permissions, further constrain the permission scope to the scope specified by the `policy` parameter.

Assume that A represents the original permissions of the CAM user, B represents the permissions that are set for the temporary key by using the `policy` parameter, and the intersection of A and B is the final valid permissions of the temporary key.

As shown in the figure below, the intersection of the CAM user permissions and the temporary permissions specified by `policy` is valid permissions:

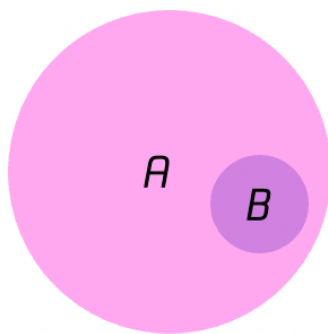


A. CAM user permissions

B. Temporary permissions specified by policy

C. Valid permissions

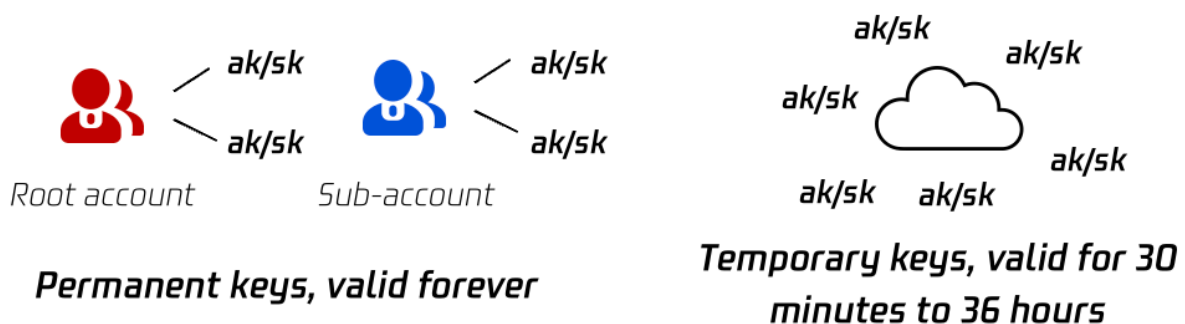
As shown in the figure below, the temporary permissions specified by `policy` are within the CAM user permissions, and therefore, the temporary permissions specified by `policy` are valid permissions.



A. CAM user permissions

B. Temporary permissions specified by policy (valid permissions)

Accessing COS Using a Temporary Key



A temporary key consists of a SecretId, a SecretKey, and a token. Multiple temporary keys can be generated for each root account or sub-account. Compared with permanent keys, temporary keys have shorter validity periods (30 minutes to 36 hours). Therefore, temporary keys are suitable for temporary authorization scenarios such as frontend direct upload. Compared with permanent keys, temporary keys are more secure to be distributed to untrusted users. For more information, see [Generating and Using Temporary Keys](#) and [Temporary Key Security Guide for Frontend Direct Upload to COS](#).

Initiating API requests

Similar to a permanent key, you can use a temporary key to generate a signature and enter it in the request header `Authorization` to form a signed request. Upon receiving the request, COS verifies whether the signature is valid and whether the temporary key has expired.

For the signature algorithm, see [Request Signature](#). COS provides a signature generation tool. You can also use a COS SDK to generate signatures. For more information, see [Implementing Signature in SDK](#).

Using an SDK development tool

After installing an SDK development tool, you can use a temporary key to initialize the user identity information. In addition, you can use a temporary key (SecretId, SecretKey, and token) to initialize COSClient and directly use an SDK to perform operations such as upload and download without generating signatures. For more information on how to generate a temporary key, see [Generating and Using Temporary Keys](#).

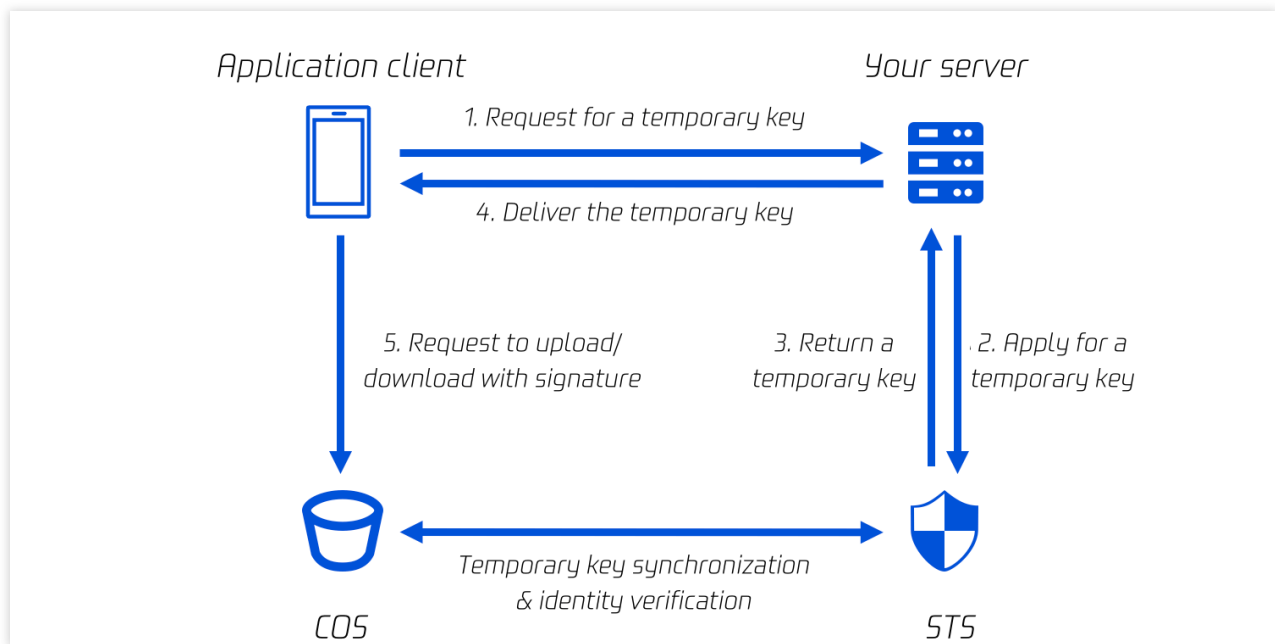
The following is an example of the corresponding Java SDK code. For demos of other languages, see the corresponding quick start documentation in [SDK Overview](#).



```
// 1. Pass in the obtained temporary key (tmpSecretId, tmpSecretKey, sessionToken).
String tmpSecretId = "SECRETID";
String tmpSecretKey = "SECRETKEY";
String sessionToken = "TOKEN";
BasicSessionCredentials cred = new BasicSessionCredentials(tmpSecretId, tmpSecretKe
// 2. Set the bucket region. For abbreviations of COS regions, please visit https:/
// `clientConfig` contains the set methods to set region, HTTPS (HTTP by default),
Region region = new Region("COS_REGION");
ClientConfig clientConfig = new ClientConfig(region);
// 3. Generate a COS client.
COSClient cosClient = new COSClient(cred, clientConfig);
```

Use Cases of Temporary Keys

Temporary keys are used to authorize third parties to temporarily access COS. For example, when a user develops a client app and stores data in a COS bucket, it is unsafe to store the permanent key directly on the app client, but the client needs to be granted the upload and download permissions. For this scenario, you can use a temporary key.



As shown in the figure above, the user has developed the app client, and the permanent key is stored in the user server. The following steps are required to use a temporary key for frontend direct upload:

1. The app client requests a temporary key from the user server for data upload and download.
2. Using a permanent key, the user server requests a temporary key from the STS server.
3. The STS server returns the temporary key to the user server.
4. The user server delivers the temporary key to the app client.
5. The app client uses the temporary key to generate a signed request to request the COS for data upload and download.

Temporary keys are applicable to frontend direct data upload scenarios. You can use temporary keys by referring to the following best practices:

[Direct Upload for Web](#)

[Direct Upload for WeChat Mini Program](#)

[Direct Upload for Mobile Apps](#)

Accessing COS Using a Pre-Signed URL

Last updated : 2024-03-25 15:33:39

COS supports object upload and download using pre-signed URLs, which are signed links with signatures embedded. You can control the effective time of a pre-signed URL based on the validity period of the corresponding signature. You can use pre-signed URLs to download objects, obtain temporary URLs for sharing files and folders temporarily, or set a long signature validity period to obtain long-term URLs for sharing files for a long time. For more information, see [\[Sharing Files\]\(#Sharing Files\)](#).

You can also use pre-signed URLs to upload objects. For more information, see [\[Uploading Files\]\(#Uploading Files\)](#).

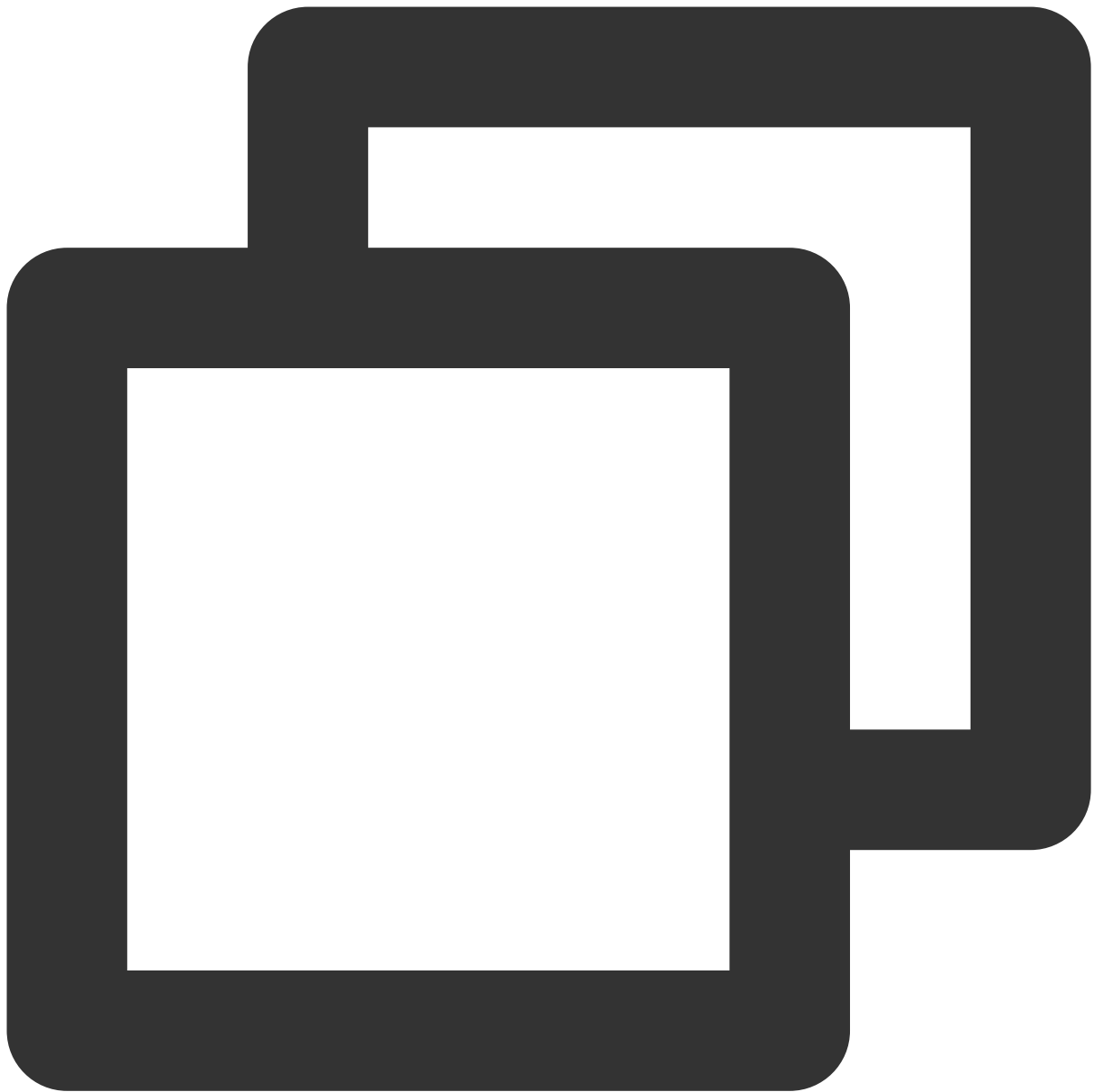
Sharing Files (Downloading Files)

COS supports object sharing. You can use pre-signed URLs to share files and folders with other users for a limited time. A pre-signed URL is in fact an object URL concatenated with a signature. For the signature generation algorithm, see [Request Signature](#).

A bucket is set to Private Read by default. If you download an object by using the object URL directly, an access failure message will be displayed. Instead, you can concatenate a valid signature to the object URL to obtain a **pre-signed URL**. The signature carries identity information, and therefore, you can successfully download the object by using the pre-signed URL.

Note:

If you need to use a permanent key to generate a pre-signed URL, you are advised to limit the permission of the permanent key only to uploads and downloads to avoid risks. In addition, the validity period of the generated URL must be set to the shortest period required to complete the current upload or download operation. This is because the request will be interrupted when the validity period of the specified pre-signed URL expires. In addition, the checkpoint restart is not supported, and therefore the failed request needs to be re-executed after a new URL is applied for.



```
// Object URL
https://test-12345678.cos.ap-beijing.myqcloud.com/test.png

// Pre-signed URL (object URL concatenated with a signature value)
https://test-12345678.cos.ap-beijing.myqcloud.com/test.png?q-sign-algorithm=sha1&q-
```

The following are a few methods of file sharing. In these methods, signatures are automatically generated for you and are concatenated to object URLs to generate temporary links that can be used directly for download and preview.

Obtaining temporary links quickly (valid for 1-2 hours)

You can quickly obtain temporary links of objects via the console or COSBrowser.

Console (web page)

1. Log in to the [COS console](#), click a desired bucket name, go to the file list page, and click **Details** corresponding to a target object.

<input type="checkbox"/>	Object Name	Size	Storage Cl...	Modification ...
<input type="checkbox"/>	1.txt	0.00B	STANDARD	2022-11-02 14:46

2. On the object details page, copy the temporary link, which is valid for 1 hour.

Information

Object Name

1.txt

Object Size

0.00B

Modification Time

2022-11-02 14:46:32

ETag

"d41d8cd98f00b204e9800998ecf8427e"

Specified Domain*(i)*

Default Endpoint

Object Address*(i)*

https://examplebucket-125...cos.ap-guangzhou.myqcloud.com/1.txt

How do I preview files directly in the browser instead of downloading them? You need to configure the correct Content-Type for

After the object address is accessed, there will be request and traffic charges. Please check the details of the charges [Billing](#)

Temporary Link*(i)*

Copy Temporary Link Download Objects Refresh

The temporary link carries the signature parameter, and the temporary link can be used to access the object during the validity period (2022-11-02 15:48:40).

Be sure to avoid leaking the temporary link, otherwise your objects may be accessed by other users.

COSBrowser (client)

For operation details, see [Generating file URL](#). Temporary links are valid for up to 2 hours when obtained by using root account keys and up to 1.5 days when obtained by using sub-account keys.

Obtaining temporary links with custom validity periods

Using the signature tool

Applicable scenario: users unfamiliar with programming

Follow the steps below:

1. File link: log in to the [COS console](#) and obtain the object address without a signature in the object details.
2. Obtain SecretId and SecretKey from [API Key Management](#).
3. Click **COS Signature Tool** to obtain the signed URL.

The effective time can be set by second, minute, hour, or day.

Using an SDK to obtain pre-signed URLs in batches

Applicable scenario: users with basic programming skills need to obtain temporary links in batches

Temporary links obtained via the console and COSBrowser have a short validity period. If you need temporary links with longer validity periods, use an SDK to generate pre-signed URLs and control the URL validity periods by specifying the signature validity period. For the pre-signed URL generation method, see the directions for an SDK in the programming language that you are familiar with in [Download via Pre-Signed URL](#).

You can use a temporary key or a permanent key to generate a pre-signed URL. The difference between the two methods is that the validity period of a temporary key is up to 36 hours but a permanent key does not expire, which indirectly affects the validity period of the pre-signed URL.

Using a permanent key to generate a pre-signed URL (any validity period)

A permanent key does not expire. The validity period of a pre-signed URL depends on the signature validity period you set. You can call the SDK's pre-signed URL directly. The operation steps are as follows:

1. Enter information such as `secret_id` , `secret_key` , and `region` to initialize the client.
2. Enter your bucket name, object name, and signature validity period to generate a pre-signed URL with a custom validity period. See the documentation for the SDK in a desired language for details:

Android SDK	C SDK	C++ SDK	.NET SDK
Go SDK	iOS SDK	Java SDK	JavaScript SDK
Node.js SDK	PHP SDK	Python SDK	Mini Program

Using a temporary key to generate a pre-signed URL (valid for less than 36 hours)

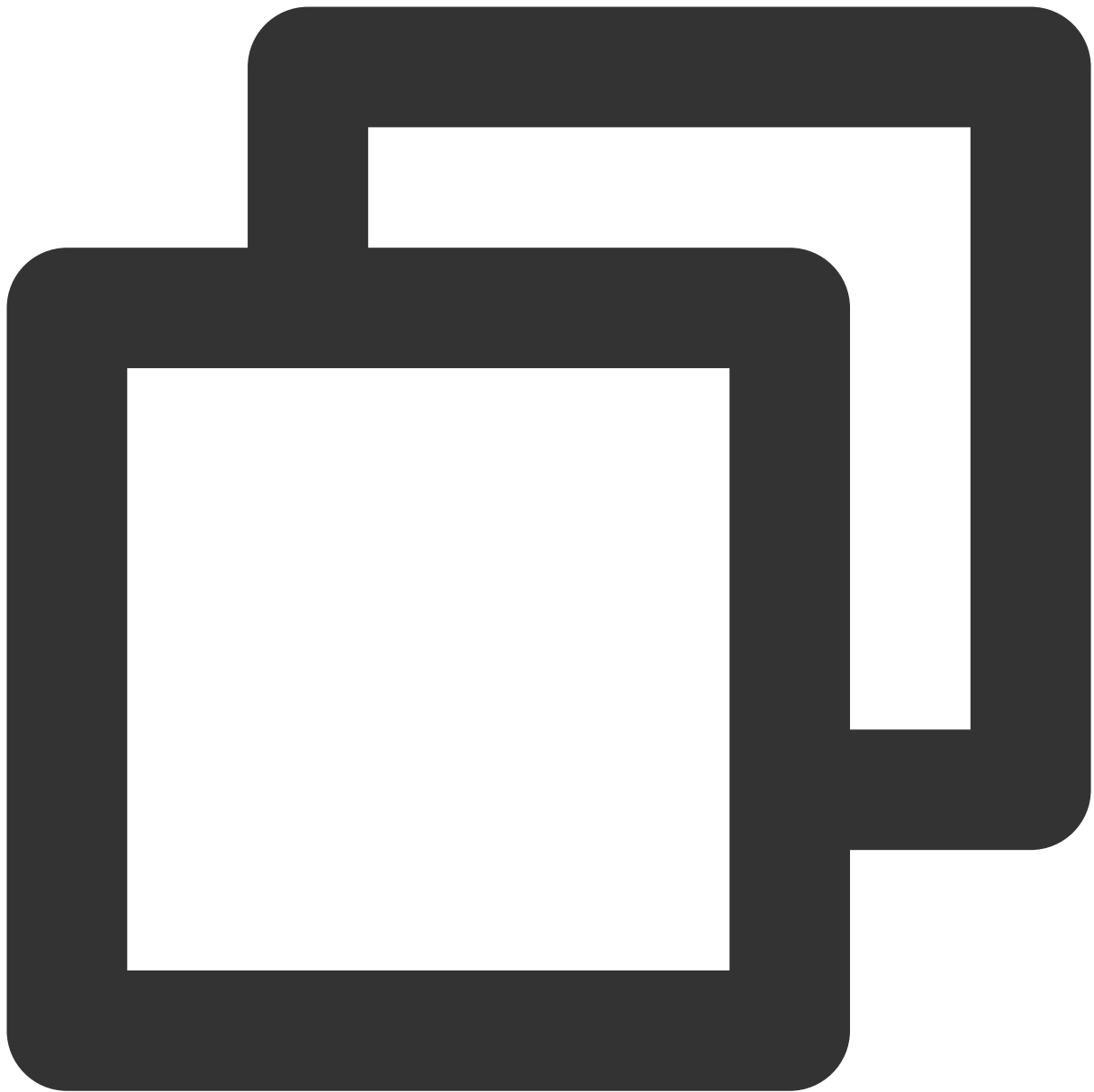
In frontend direct upload scenarios, temporary keys are often required. For more information about temporary keys, see:

[Accessing COS Using a Temporary Key](#)

[Generating and Using Temporary Keys](#)

[Temporary Key Security Guide for Frontend Direct Upload to COS](#)

The maximum validity period of a temporary key is 36 hours. The validity period of a pre-signed URL is either the validity period of the signature or that of the temporary key that you set, whichever is smaller. For example, if you set the validity period of the signature to X and the validity period of the temporary key to Y, the actual effective time of the link is T:



```
T=min(X,Y); because X is less than or equal to 36, we can infer that T is also less
```

To use a temporary key to generate a pre-signed URL, perform the following steps:

1. [Get a temporary key](#).
2. Once the temporary key is obtained, you can use a function similar to the permanent key to generate a pre-signed URL. Note that to initialize the client with a temporary key, you need to enter the SecretId, SecretKey, and token, and carry the parameter `x-cos-security-token`. See the documentation for the SDK in a desired language for details:

[Android SDK](#)[C SDK](#)[C++ SDK](#)[.NET SDK](#)

Go SDK	iOS SDK	Java SDK	JavaScript SDK
Node.js SDK	PHP SDK	Python SDK	Mini Program

Sharing a Folder

Folders are a special type of object. You can share a folder via the console or COSBrowser. For more information, see [Sharing a Folder](#).

Uploading Files

If you want any third party to be able to upload an object to your bucket, but you don't want them to use CAM accounts or temporary keys, signatures can be provided by pre-signed URLs for temporary upload operations. Anyone who receives a valid pre-signed URL can upload an object.

Note:

If you need to use a permanent key to generate a pre-signed URL, you are advised to limit the permission of the permanent key only to uploads and downloads to avoid risks. In addition, the validity period of the generated URL must be set to the shortest period required to complete the current upload or download operation. This is because the request will be interrupted when the validity period of the specified pre-signed URL expires. In addition, the checkpoint restart is not supported, and therefore the failed request needs to be re-executed after a new URL is applied for.

Method 1. Generating a pre-signed URL via SDK

SDKs in various programming languages provide pre-signed URL generation methods. For more information about the methods, see [Upload via Pre-Signed URL](#). Select a method according to the programming language that you are familiar with.

Method 2. Manually constructing a pre-signed URL

A pre-signed URL is in fact an object URL concatenated with a signature. Therefore, you can use an SDK or signature generation tool to generate a signature and concatenate the object URL and the signature to form a pre-signed URL for object upload. However, this method is generally not recommended because of the complexity of the signature generation algorithm.

Accessing COS Anonymously

Last updated : 2024-03-25 15:33:39

COS buckets are private by default. Identity verification is required for any access to COS, and object URLs used to access COS must carry signature information. However, if resources (buckets, objects, and folders) are configured with the Public Read permission, anonymous access is allowed, and users can use object URLs to download resources directly.

According to the permission application scope, COS allows users to set the Public Read permission at the bucket, object, or folder level.

Setting the Permission of a Bucket to Public Read

If the permission of a bucket is set to Public Read/Private Write, all objects in the bucket can be accessed by anonymous users. For the setting method, see [Setting Access Permission](#).

Setting the Permission of an Object to Public Read

If the permission of a specified object is set to Public Read/Private Write, the object can be accessed via the object URL. For the setting method, see [Setting Object Access Permission](#).

Setting the Permission of a Folder to Public Read

If the permission of a folder is set to Public Read/Private Write, all files in the folder can be accessed by anonymous users. For the setting method, see [Setting Folder Permissions](#).

Public Read Permission Evaluation Mechanism

For the COS permission evaluation mechanism, see [Access Policy Evaluation Process](#). If the Public Read permission settings for buckets, folders, and objects conflict, the priorities are as follows:

The ACL of an object has the highest priority. If the ACL of an object is inherited, the ACL of the corresponding folder applies. If the ACL of a folder is inherited, the ACL of the corresponding bucket applies.

Using CDN to Accelerate Access

CDN Acceleration Overview

Last updated : 2024-03-25 15:33:39

Content Delivery Network (CDN) can be used to accelerate mass download/deliver content stored in a COS bucket, which is ideal when the same content needs to be downloaded repeatedly. The origin-pull authentication feature allows CDN to accelerate content stored in a private-read bucket. The CDN authentication feature allows only the authorized users to download content to avoid data security risks and unnecessary traffic costs.

Note:

After you enable the CDN acceleration domain name, data downloads and access through it will generate CDN origin-pull traffic and CDN traffic. For more information, see [Traffic Fees](#).

CDN

CDN overview

CDN is a layer in the internet ecosystem, consisting of high-performance edge nodes distributed around the world. These nodes store your content according to the caching rules. When a user requests content, the request will be routed to the edge node closest to the user to speed up access and improve availability.

CDN involves caching and origin-pull. When a user accesses a URL, if the requested content is not cached on the edge node, or the cached content has expired, the content will be pulled from the origin.

Use cases

Low latency and fast downloads are required.

GB- to TB-level data needs to be transferred across regions, countries, or continents.

The same content needs to be downloaded frequently.

Security options

Origin-pull authentication: If the requested content is not cached on the edge node, the content will be pulled from the origin. If COS is used as the origin and origin-pull authentication is enabled, the CDN edge node will use a special service identity to access the COS origin to obtain and cache the data in the private bucket.

CDN service authorization: You can authorize the CDN service so that CDN edge nodes can use a special service identity to access the COS origin and pull content from it.

CDN authentication: When a user accesses an edge node to acquire cached data, the edge node verifies the authentication field in the access URL based on the authentication configuration rules. This prevents unauthorized access and hotlinking, thereby improving the security and reliability of the data cached in the edge node.

Access Nodes of COS

Definition of access node

An access node is a domain name assigned to a bucket according to the bucket's region and name. You can access data stored in the bucket using this domain name.

If the static website feature is enabled, you will be provided a static website access node, which can be configured responses that are different from that of the default access node.

Access nodes

Access node: When a bucket is created, COS will assign an XML access node to the bucket in the format of

`<BucketName-APPID>.cos.<Region>.myqcloud.com` , which can be accessed using RESTful APIs. You can access the node and refer to the [API Documentation](#) to configure the bucket, or upload/download objects.

Static website node: You can enable the static website feature on the **Basic Configurations** page of the bucket in the console. After this, you will be provided an access node formatted as `<BucketName-APPID>.cos-website.<Region>.myqcloud.com` . You can configure special index pages (IndexPage), error pages (ErrorPage), and redirects for your static website, which allows only object downloads. Users can obtain the content using the static website domain name.

Access permissions

Public-read: If a bucket is set to public-read, everyone can access the bucket using its domain name. If you use a public-read bucket as the origin, you can enable CDN acceleration directly without enabling CDN authentication and origin-pull authentication.

Private-read: If a bucket is set to private-read, you can create access policies to manage who can access the bucket and manage CDN authorization. If you use a private-read bucket as the origin and enable origin-pull authentication but not CDN authentication, unauthorized users can access the bucket via CDN. Therefore, **you are advised to enable both origin-pull authentication and CDN authentication for private-read buckets to ensure the data security.**

Accelerating Access to COS Using CDN

You can use a custom domain name that has obtained an ICP filing number and use a COS bucket as the origin. This allows you to accelerate access to objects in the bucket using the custom CDN acceleration domain name.

Note:

A Tencent Cloud CDN acceleration domain name does not provide an IP address by default. To know more about the DNS of the domain name, run the `dig` command for query.

Public-read buckets

If a bucket is set to public-read and COS is used as the origin for CDN pulling, you don't need to enable origin-pull authentication and CDN edge nodes can obtain and cache objects stored in the COS bucket.

You can still protect your objects in the bucket **to some extent** by enabling [Authentication Configuration](#) in the CDN console. Regardless of whether this feature is enabled, users who know the bucket access domain name can access all objects in the bucket. Whether users can access the public-read bucket in different CDN authentication configurations is described in the following table:

CDN Authentication	Access with CDN Acceleration Domain Name	Access with COS Domain Name	Use Case
Disabled (default)	Yes	Yes	Allowing all public access to the entire website via the CDN or origin
Enabled	URL authentication is required	Yes	Enabling hotlink protection for access via the CDN but not the origin server (not recommended)

Private-read buckets

If a bucket is set to private-read (default) and COS is used as the origin for CDN pulling, CDN edge nodes **cannot obtain and cache any objects**. Therefore, you need to add the CDN service identity to the bucket policy and authorize the identity to perform the following operations:

GET Object: downloads an object.

HEAD Object: queries object metadata.

OPTIONS Object: configures a CORS preflight request

You can authorize the identity quickly in either the [CDN console](#) or the [COS console](#) by clicking **Add CDN Service Authorization**. Then, enable **Origin-pull Authentication**. In this way, CDN edge nodes can use the service identity to access COS objects.

Note:

If the bucket is set to private-read, you must add the service authorization and enable origin-pull authentication; otherwise, access to COS will be denied.

A CDN edge node will generate a service account for each root account. Therefore, the account authorization is only valid for the root account to which the acceleration domain name belongs. If the acceleration domain name is bound to another account, access will be denied.

After you added the CDN service authorization and enabled origin-pull authentication, CDN edge nodes can obtain and cache data. Therefore, you are advised to enable [Authentication Configuration](#) if you need to protect private data stored in the bucket. Whether users can access the private-read bucket in different CDN authentication configurations is described in the following table:

CDN	Access with CDN	Access with	Use Case
-----	-----------------	-------------	----------

Authentication	Acceleration Domain Name	COS Domain Name	
Disabled (default)	Yes	COS authentication is required	Allowing direct access to CDN domain names to protect the content on the origin
Enabled	URL authentication is required	COS authentication is required	Securing access comprehensively (hotlink protection for CDN authentication is supported)

Setting CDN Acceleration

Last updated : 2024-03-25 15:33:39

Use Cases

Low latency and fast downloads are required.
GB- to TB-scale data needs to be transferred across regions, countries, or continents.
The same content needs to be downloaded frequently.

Note:
If the download request is from a Tencent Cloud VPC (for example, using a CVM to access a COS bucket), we recommend you use the COS domain directly. If you use a custom CDN acceleration domain, you will need to access the CDN node over a public network, which incurs additional fees such as CDN origin-pull fees and CDN traffic fees.

Notes

For information on domain name definition, CDN origin-pull authentication, and CDN authentication configuration, see [CDN Acceleration Overview](#).

CDN origin-pull authentication and CDN authentication configuration affect how custom CDN acceleration domain names and COS domain names access origin buckets. Please find the details in the table below.

Bucket access permission	CDN origin-pull authentication	CDN authentication	Origin can be accessed via custom CDN acceleration domain name	Origin can be accessed via COS endpoint	Scenarios
Public read	Disabled	Disabled	Yes	Yes	Site-wide public access
Public read	Enabled	Disabled	Yes	Yes	Not recommended
Public read	Disabled	Enabled	URL authentication is required	Yes	Not recommended
Public read	Enabled	Enabled	URL authentication is required	Yes	Not recommended

Private read + CDN service authorization	Enabled	Enabled	URL authentication required	COS authentication required	Full-linkage protection
Private read + CDN service authorization	Disabled	Enabled	URL authentication is required	COS authentication is required	Not recommended
Private read + CDN service authorization	Enabled	Disabled	Yes	COS authentication is required	Origin protection
Private read + CDN service authorization	Disabled	Disabled	No	COS authentication is required	Not recommended
Private read	Disabled	Enabled or disabled	No	COS authentication is required	CDN is unavailable

Note:

Origin protection is useful in cases where your data cached on CDN edge nodes may be maliciously pulled due to lack of CDN authentication. Therefore, we strongly recommend you enable CDN authentication to mitigate data security issues.

Setting CDN Acceleration

You can bind a custom domain name with a bucket in the COS console. After that, you can enable CDN acceleration for the custom domain name to speed up access to the bucket through the custom domain name. When binding a custom domain name with a bucket, you need to add a CNAME record to it at your domain name service provider.

Note:

Currently, you must activate the CDN service to use a custom acceleration domain name in COS.

1. For content delivery in the Chinese mainland, ICP filing is required. You are not required to do so through Tencent Cloud though.
2. For content delivery outside the Chinese mainland, ICP filing is not required, but note that your data and operations in Tencent Cloud still need to comply with local laws and regulations as well as [General Service Level Agreements](#).

Prerequisites

1. Domain name registration. You can register your domain name with [Tencent Cloud](#) or another service provider.
2. ICP filing application. For content delivery in the Chinese mainland, complete ICP filing.

Directions

Note:

You can add a custom domain name and enable CDN acceleration in both COS console and CDN console. For more information on how to do so in the CDN console, see [Connecting Domain Names](#).

1. Select a bucket

Log in to the [COS console](#), click **Bucket List** on the left sidebar, and click the name of a bucket that you want to enable CDN acceleration.

2. Add a custom CDN acceleration domain name

Click **Domains and Transfer > Custom CDN Acceleration Domain**. Under the **Custom CDN Acceleration Domain** configuration item, click **Add Domain**.

Domain Name: Enter the target custom domain name (such as `www.example.com`). Make sure that an ICP filing has been obtained and a CNAME record has been configured at the DNS service provider for the entered domain. For more information, see [CNAME Configuration](#). If the custom CDN acceleration domain you are connecting is in the following situations, you need to verify your domain ownership as instructed in [Domain Name Ownership Verification](#).
The domain name is being connected for the first time.

The domain name has been connected by another user.

The domain name is a wildcard domain name.

Acceleration Region: CDN acceleration is supported for regions in the Chinese mainland, outside the Chinese mainland, and globally, with global acceleration for buckets across all regions.

Origin Server Type: **Default Endpoint** is selected by default. If a static website is enabled for a bucket that serves as an origin, and you want to accelerate the static website, select **Static Website Endpoint**. For more information, see [CDN Acceleration Overview](#).

Authentication: Enable origin-pull authentication. For private-read buckets, enable origin-pull authentication to protect the origin.

Note:

For private-read buckets, if both origin-pull authentication and CDN service authorization are enabled, signature is not required for accessing the origin via CDN, and cached resources in CDN will be distributed on the public network, which will affect the data security. Therefore, we recommend that you enable CDN authentication.

Enabling origin-pull authentication

Origin-pull authentication is used to verify the service identity of the CDN edge node so as to prevent unauthorized access. Find the details below.

Public-read bucket: The CDN edge node can access the bucket without any authorization, so there is no need to enable origin-pull authentication.

Private-read bucket: A CDN edge node needs to go through origin-pull authentication to get its service identity verified before it can access the objects in the bucket. Click to enable **Origin-pull Authentication**.

After enabling **Origin-pull Authentication**, click **Save** on the right. In about five minutes, the added custom domain name and CDN acceleration will be deployed.

Enabling CDN authentication

Note:

After CDN acceleration is enabled for the custom domain name, anyone can access the origin via the domain name. Therefore, if you need to keep your data private, be sure to protect your data in the origin by enabling CDN authentication.

After the custom domain name is deployed, a CDN authentication configuration link will appear in the CDN authentication column. Click **Set** to directly enter the CDN console for CDN authentication configuration. For detailed directions, see [Authentication Configuration](#).

Automatic CDN Cache Purge: After this feature is enabled, when a file in the COS bucket is updated, the CDN cache will be automatically purged. You can go to the **Function Service** section in the COS console to configure this feature as instructed in [Setting CDN Cache Purge](#).

HTTPS Certificate: To add an HTTPS certificate for the custom domain name, go to the [CDN console](#).

3. Resolve the domain name

After the custom domain name is added to the CDN, the system will automatically assign you a CNAME domain name suffixed with `.cdn.dnsv1.com`. You need to complete the CNAME configuration at the domain name service provider. For more information, see [CNAME Configuration](#).

Note:

You will not be able to directly access the CNAME domain name.

4. Disable the feature

After the above steps are completed, you can use the custom domain name to accelerate access to resources in the bucket. You can disable it in the following ways:

On the **Custom CDN Acceleration Domain** page, click **Edit** to change **Status** from **online** to **offline**, click **Save**, and wait for about 5 minutes for the deployment to complete. After that, the status of the custom acceleration domain name will be changed to **Deactivated** in the CDN console.

Note:

You can delete a custom domain name only after changing its status to **Offline**. To disable or delete a CDN domain name, go to the [CDN console](#). For more information, see [Domain Name Operations](#).

Single-Connection Bandwidth Limit

Last updated : 2024-03-25 15:33:39

Single-Connection Bandwidth Limit

COS supports traffic control for file uploads and downloads to guarantee normal network bandwidth for your other applications. You can add the `x-cos-traffic-limit` parameter in a request for [PutObject](#), [PostObject](#), [GetObject](#), or [UploadPart](#) and set a speed limit value. Then, COS will control the network bandwidth for the request accordingly.

Instructions

You can put a traffic limit on a request by specifying the `x-cos-traffic-limit` in your PUT Object, POST Object, GET Object, or Upload Part request. This parameter can be used as a request header, request parameter, or a HTML form field for POST Object requests.

The value of the `x-cos-traffic-limit` parameter must be a number and the unit is bit/s by default.

The speed limit value range is 819200–838860800, i.e, 100 KB/s–100 MB/s. If the range is exceeded, a 400 error will be returned.

Note:

1 MByte = 1,024 KByte = 1,048,576 Byte = 8,388,608 bit.

API Use Case

The following is a sample request for the simple upload API with a speed limit of 1048576 bit/s (128 KB/s):



```
PUT /exampleobject HTTP/1.1
Host: examplebucket-1250000000.cos.ap-beijing.myqcloud.com
Content-Length: 13
Authorization: q-sign-algorithm=sha1&q-ak=AKID8A0fBVtYFrNm02oY1g1JQQF0c3JO***&q-si
x-cos-traffic-limit: 1048576
```

Batch Operation

Managing Batch Operation Jobs

Last updated : 2024-03-25 15:33:39

This document describes how to manage batch operation jobs in the COS console.

Filtering Jobs

You can view the list of the batch operation jobs created in the last 90 days using the [List Jobs](#) API. The list contains information of each job, such as job ID, description, priority, status, and execution progress. You can filter them by job status in the list to display the ones in the same status or filter them by job description or ID in the console.

Querying Job Status

If you need more information, you can get all the information of a job using the [DescribeJob](#) API. This API will return information such as the task configuration of the specified job, object inventory information, and job report.

Setting Job Priority

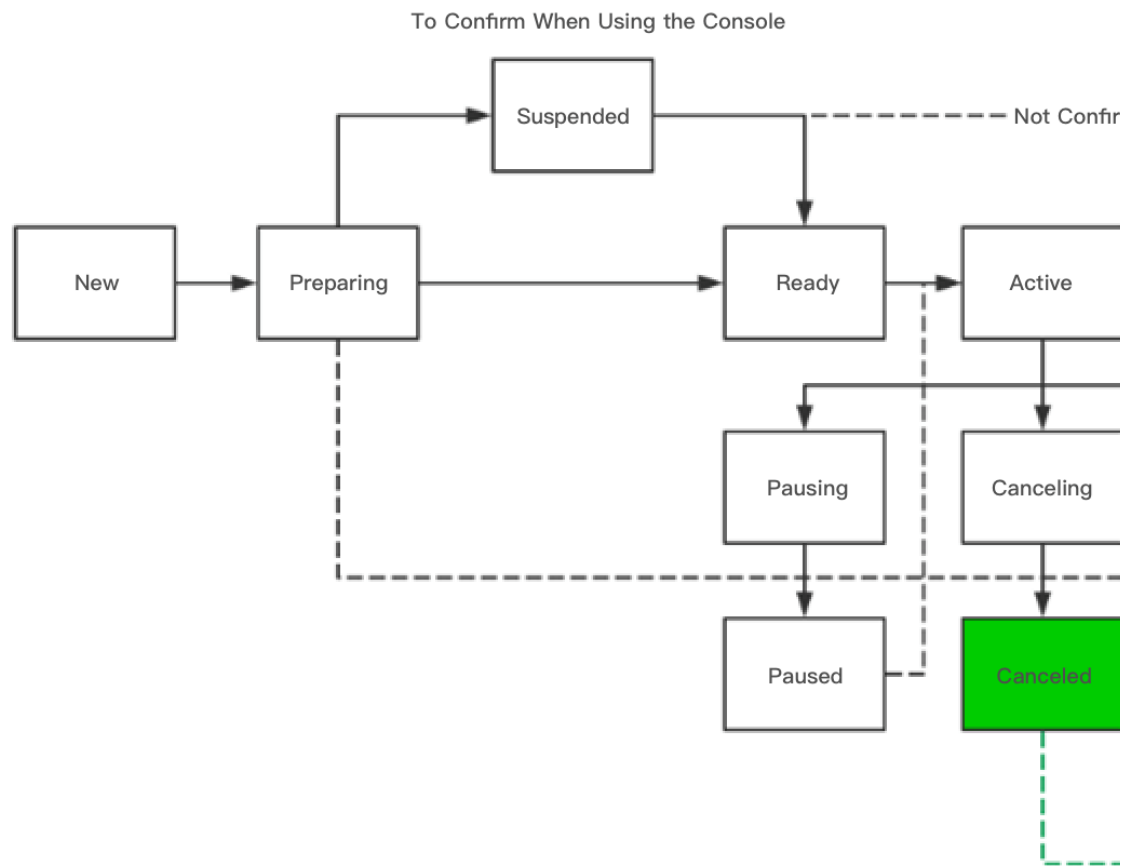
You can set priority for your batch operation jobs, and COS will execute them based on priority, i.e., executing jobs of a higher priority first. Priority is expressed as an integer; the greater the value, the higher the priority. You can modify the priority during job execution. If a higher-priority job needs to be added, you can run it first by pausing the lower-priority job.

Note:

Although higher-priority jobs generally take precedence over lower-priority ones, priority is not a criterion for sequential execution. If you want to run batch operation jobs sequentially, please monitor the execution status of each job on your own and start them manually.

Job Status

Job status changes as the execution proceeds, as shown below:



The specific meaning of each job status is as follows:

Job Status	Description	Next Status
New	When a job is created, its status will be "New".	It can change to "Preparing", meaning that COS starts to parse the inventory in the job.
Preparing	When COS starts to parse the configured inventory in the batch operation job, the job status will change to "Preparing".	It can change to "Ready" or "Suspended". The former means that COS has finished parsing the job configuration information and is going to perform the specified tasks on the objects in the inventory based on the configuration, while the latter means that COS has finished parsing the job configuration information but is still waiting for your confirmation before it proceeds to perform the specified tasks (this status appears if a batch operation job is configured in the console).

Suspended	A batch operation job pending your confirmation is in the "Suspended" status, which appears if the job is created in the console. After your confirmation, COS will start executing the job, and the status will change accordingly and cannot revert to "Suspended".	When you confirm to execute the job, the job status will change to "Ready".
Ready	When COS has finished parsing the object inventory and job configuration information in your batch operation job and is going to perform tasks, the job status will change to "Ready".	It can change to "Active", at which point COS starts executing the job. If a higher-priority job is running, COS will keep the job status as "Ready" until the job status of the higher-priority job changes to "Complete".
Active	When COS is performing tasks on the objects in the inventory based on the configuration information, the job status will be "Active". You can query job progress in the console or by calling the DescribeJob API.	It can change to "Complete", "Failing", "Pausing", or "Canceling" if your job succeeds, fails, or is paused or canceled.
Pausing	After COS pauses an ongoing job and before the status changes to "Paused", the job status in between will be "Pausing".	It can be followed by "Paused" when COS has successfully paused the ongoing job.
Paused	If a higher-priority job is created, the status of the ongoing job will change to "Paused".	If the higher-priority job is completed, fails, or is pending conformation, the job in "Paused" status can automatically shift to "Active" status.
Complete	When the batch operation job has successfully completed the tasks on all the objects in the inventory or fails, the job status will change to "Complete". If job report generation is configured, COS will deliver the report to your specified bucket when the status changes to "Complete".	"Complete" is a final status, i.e., the job will not change to any other status.
Canceling	After COS cancels an ongoing job and before the status changes to "Canceled", the job status in between will be "Canceling".	It can be followed by "Canceled" when COS has successfully canceled the ongoing job.

Canceled	When a batch operation job is successfully canceled, the job status will change to "Canceled". At this point, you cannot make any modification to the job status.	"Canceled" is a final status. i.e., the job will not change to any other status.
Failing	The "Failing" status comes before "Failed".	It can change to the "Failed" status.
Failed	After a job fails, the job status will change to "Failed". For more information, see Tracking Job Failure .	"Failed" is a final status, i.e., the job will not change to any other status.

Tracking Job Failure

If a problem arises during job execution, such as trouble in parsing the object inventory, the batch operation job will fail, and COS will return the corresponding error code and cause. You can call the [DescribeJob](#) API or view the job report to get the cause of failure and other information.

COS sets a task failure threshold for each batch operation job to avoid frequent task failures. If a job involves more than 1,000 tasks, COS will monitor the task failure rate, i.e., proportion of failed tasks to all tasks executed. If the rate exceeds the threshold of 50% at any moment, COS will terminate the job and return the failure message. You can check the cause why this happens (e.g., the object inventory contains large amounts of information on non-existing objects). Then, you can fix the problem accordingly and create a job again.

Note:

COS executes batch operation jobs in an async manner and does not necessarily perform tasks in the same order that the objects are listed in the inventory. Therefore, you cannot determine which object is being operated on according to the order in the object inventory and whether a task is successful or fails. However, you can get the information on successful or failed tasks from the job report.

Job Report

You can configure whether to output a job report when creating a job. If yes, COS will output a job report when the job succeeds, fails, or is canceled. You can view the information on all successful or failed tasks in the job report.

A job report contains information such as the configuration parameters and execution status of the specified task as well as name and version ID of the objects operated on, task status codes, and error description.

Overview

Last updated : 2024-03-25 15:33:39

With the batch operation feature in COS, you can specify an operation on a specified list of objects in a bucket. To do so, you have two options: either create an inventory of the objects via the inventory feature or list the desired objects in a CSV file as formatted in the inventory. Those objects will then be batch operated as specified in COS.

For more information, see [Inventory Overview](#).

At present, the batch operation feature only supports the following operations:

[Batch copying objects](#)

[Batch restoring archived objects](#)

You can use the batch operation feature in the COS console. For more information, see [Batch Operation](#).

How It Works

To perform a batch operation, you need to create a batch operation job first, which contains all the information needed to perform the specified operation on the object list. You can use an inventory as an object list.

After you provide an inventory file and start the created batch operation job, the batch operation feature will execute the specified operation on the objects in the inventory sequentially. During job execution, you can monitor the execution status in the COS console or choose to output a job report after the job is completed. The job report provides details about each operation in the job.

Note:

The batch operation feature is only applicable to objects in the current bucket. If you want to batch operate on the objects in another bucket, please enable the batch operation feature for that bucket.

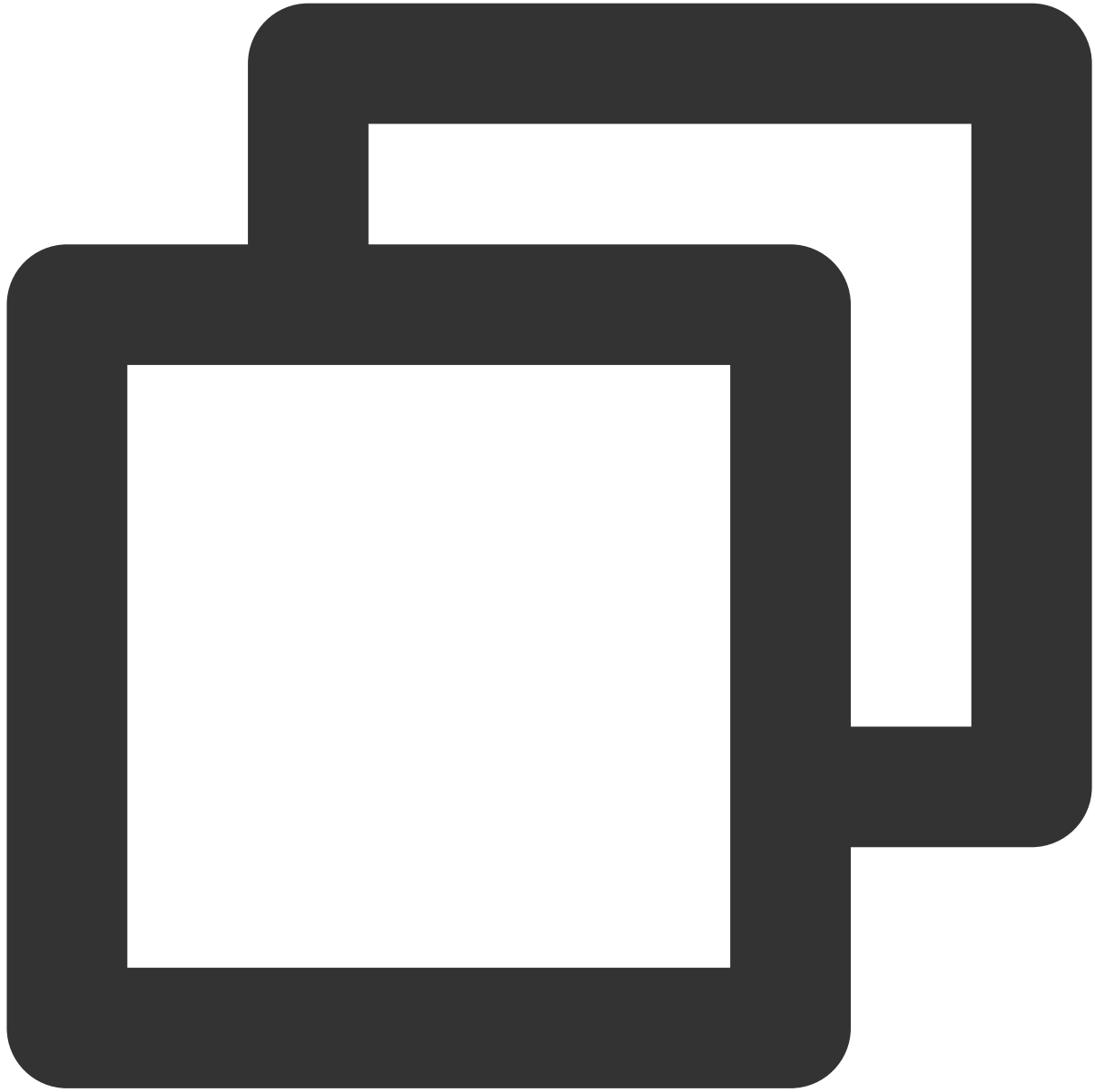
Object Inventory

An object inventory is a list of all the objects to be operated on. To create a batch operation job, you need to provide an object inventory first to tell COS what objects it should operate on. You need to put the object inventory file in the bucket and provide information such as the filename, ETag, and VersionID (if applicable). You can create an object inventory in the following two ways:

COS inventory feature: This feature outputs an object inventory in CSV format. For more information, see [Inventory Overview](#). If your object inventory contains version ID information, COS will batch operate on the objects with the corresponding version ID.

CSV file configuration: Every row in the CSV file must contain the bucket name and the name and version ID (if versioning is enabled for the bucket) of an object for batch operation. If versioning has never been enabled, you can

skip the version ID information. The CSV file can be configured as follows:



```
examplebucket-appid, exampleobject, PZ9ibn9D5lP6p298B7S9_ceqx1n5EJ0p  
examplebucket-appid, exampleobject, jbo9_jhdPEyB4RrmOxWS0kU0EoNrU_oI
```

Note:

If versioning is or was once enabled for your bucket, and you want to perform a batch operation on the specified version of the object, you must provide object version ID information in the object inventory.

If versioning is or was once enabled for your bucket, but you don't specify a version ID in the inventory, COS will operate on the latest version of the object by default.

If you uploaded an object with the same name as an object to be operated on before creating a job, COS will operate on the object on the latest version by default rather than the version when the object inventory was created. To avoid this issue, you can enable versioning and specify the version ID in the object inventory.

An object inventory can contain all the objects in a bucket. However, note that it takes longer to operate on a large number of objects.

Batch Operation Job

This section describes how to create a batch operation job and how the system will respond after creation.

When creating a batch operation job, you need to provide the following information:

Type	Description
Operation	You need to specify which operation is to be performed on objects. Corresponding parameters can be configured for each operation, and COS will perform the operation on the objects in the inventory sequentially as configured.
Object inventory	An object inventory is a list of all the objects to be operated on. You can create an object inventory through the inventory feature. For more information, see Inventory Overview . You can also list desired objects in a CSV file as formatted in the inventory.
Priority	You can set priority to identify the precedence of the current batch operation job over other jobs. Job priority does not directly determine the order in which jobs will be completed. If you want to control the order of multiple jobs, you need to check the job execution status on your own and start the next job after the current one is completed.
Rule permission	After creating a batch operation job, you need to make sure that your account has the corresponding IAM permission to perform the operation. For example, if you have created a batch operation job to execute <code>PUT Object-copy</code> , you should make sure that your account has the <code>Get Object</code> permission in the source bucket and the <code>PUT Object</code> permission in the destination bucket. In addition, for all batch operation jobs, you should have permission to read the object inventory and write into the job report. For more information on permission configuration, see Overview and Bucket Policy .
Job report	If you want to output a job report after a batch operation job is completed, you need to enter the corresponding parameters when creating the job, so that the system can correctly output the job report to the specified destination bucket. The required information includes the bucket to store the job report, job report format, and whether all job information should be included. The file prefix of the job report is optional.
Job description (optional)	You can enter 256-byte job description for your created batch operation job to keep track of it. The job description will be displayed in the COS console and can be used to sort or filter jobs. You can enter same job description for similar jobs (e.g., syncing and copying log data weekly) to manage them in a centralized manner.

Performing Batch Operation

Batch Copying Objects

Last updated : 2024-03-25 15:33:39

The batch copy feature is used to copy objects in the inventory, i.e., allowing you to batch copy the specified objects from the source bucket to the destination bucket in the same region or in a different region. It supports customizing parameters for the `PUT Object-copy` operation, and parameter configuration affects the metadata and storage class of the target objects. For more information, see [PUT Object-copy](#).

Restrictions

All objects to be copied must be in the same bucket.

Only one destination bucket can be configured for a batch copy job.

You need to have permission to read objects from the source bucket and write objects into the destination bucket.

The total size of the objects cannot exceed 5 GB.

Verification through ETags and server-side encryption using custom keys are not supported.

If the destination bucket does not have versioning enabled and contains an object with the same name as an object to be copied, COS will overwrite the object.

If an object in the inventory has multiple versions, only one version can be copied. If the version ID is not specified, the latest version will be copied.

Batch Restoring Archived Objects

Last updated : 2024-03-25 15:33:39

The batch restoration feature can be used to restore archived objects in the inventory. This operation supports configuring parameters for `POST Object restore`. The configuration information will affect the restoration time and expiration time of copies. For more information, see [POST Object restore](#).

Restoration is supported for ARCHIVE and DEEP ARCHIVE storage classes. For more information on these two storage classes, see [Storage Class Overview](#).

To create a batch restoration job, you need to specify the following two parameters:

Restoration mode: Standard retrieval or bulk retrieval. For more information, see [Restoring Archived Objects](#).

Copy validity period: When an archived object is restored, a temporary copy will be created, which will expire and be deleted automatically after the specified time period. For more information on the copy validity period, see [Restoring Archived Objects](#).

Notes

1. If the batch restoration job includes copies of objects that have already been restored, the validity period of these copies will be updated when the job starts, ensuring that all copies in the job have the same validity period.
2. The batch restoration job only initiates requests to restore objects. After all requests are sent, the **Batch Operation** page will display the job as completed. COS doesn't notify you when the objects have been restored, but you can configure event notifications to receive notifications. For more information, see [Event Notifications](#).

Global Acceleration

Overview

Last updated : 2024-03-25 15:33:39

The global acceleration feature provided by Tencent Cloud Object Storage (COS) utilizes a load balancing system based on Tencent's global traffic scheduling to intelligently route and parse user requests and select the optimal network linkage for nearby access. Backed by globally deployed Tencent Cloud data centers, it allows users across the world to quickly access buckets and improves application access success rate, allowing for greater business stability and a smoother user experience. In addition, COS's global acceleration also speeds up uploads and downloads.

Note:

The global acceleration feature is now generally available and supported in all public cloud regions. Using this feature incurs fees as request data transfers are accelerated via Direct Connect lines in the Tencent Cloud private network. For more pricing information, see [Product Pricing](#).

Directions

You can enable global acceleration on the COS Console or through APIs.

Using the COS console

You can enable global acceleration for your buckets in the COS console. For more information, see [Enabling Global Acceleration](#) in the console documentation.

Using RESTful APIs

You can directly use the following APIs to enable global acceleration:

[PUT Bucket Accelerate](#)

[GET Bucket Accelerate](#)

Access Endpoint Domain Names

After enabling global acceleration, you can access your COS files through two types of endpoint domain names:

Default bucket endpoint domain name: Format: `<BucketName-APPID>.cos.<Region>.myqcloud.com` .

For more information, see [Regions and Access Endpoints](#).

Global acceleration endpoint domain name: Format: `<BucketName-APPID>.cos.accelerate.myqcloud.com` .

Take the bucket `examplebucket-1250000000` in Guangzhou as an example. If you have enabled global acceleration for it, when you need to upload the file `exampleObject.txt` from Beijing to it, you can do so in the following two ways:

Use the global acceleration endpoint domain name for access: when uploading the object, you need to set the endpoint domain name to `exampleBucket-1250000000.cos.accelerate.myqcloud.com`. When you upload the object through this endpoint domain name, COS will intelligently parse your request based on your network conditions and implement nearby access. For example, it will forward your request to the Beijing access layer and then transmit it to the Guangzhou storage layer via a private network Direct Connect line to accelerate data transfer.

Use the default bucket endpoint domain name for access: when uploading the object, you need to set the endpoint domain name to `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com`. When you upload the object through this endpoint domain name, your request will be directly forwarded to the Guangzhou access layer and then the Guangzhou storage layer. In this case, long public network linkage may lead to unstable transfer.

Note:

If you use global acceleration, fees will be incurred. Therefore, we recommend that you carefully evaluate whether to use this feature with your actual business needs in mind:

1. If your business has more writes (e.g., `PUT Object`, `POST Object`, and `Multipart Upload`) than reads and uploads data to Tencent Cloud data centers from a remote region, we recommend using a global acceleration endpoint domain name.
2. If your business has more reads (e.g., `GET Object`) than writes and mainly involves file download, we recommend that you perform a comprehensive evaluation of the [CDN-based access acceleration](#) solution and select the option with the best cost performance.
3. If your business mainly involves configuration operations or file extraction, we recommend using the default bucket endpoint domain name.
4. If your business needs to access buckets over a private network or through a Direct Connect line within the same region, we recommend using the default bucket endpoint domain name.

Notes

We have outlined below some important factors to note when using a global acceleration endpoint domain name:

The global acceleration domain name will take effect 15 minutes after being enabled.

After the global acceleration domain name is enabled, the maximum bandwidth for a bucket will be allocated based on the business volume of the entire network.

After the global acceleration domain name is enabled, only requests using that domain name will be accelerated. However, the default bucket domain name can still be used.

When using a global acceleration domain name, fees will be incurred only for requests for which linkage is accelerated. For example, if you use a global acceleration domain name to upload data from Beijing to a bucket in

Beijing, the request will not incur acceleration fees as the linkage was not accelerated.

When using a global acceleration domain name, you can specify the HTTP or HTTPS transfer protocol. However, if the request is transmitted via a private network Direct Connect line, COS will choose to use the HTTPS protocol to guarantee data transfer security.

Billing Example

Uploading data or accessing a bucket by using a global acceleration endpoint domain name will incur fees calculated by the **day**. For more information, see [Billing Overview](#) and [Product Pricing](#). The following example compares billing between a global acceleration endpoint domain name and a default endpoint domain name:

Business scenario 1

A user uses COS mainly to upload video files which require a high transmission success rate. If he/she uploads 1 GB of video data daily from his/her offices in Xinjiang and Singapore to his/her bucket in Guangzhou, then 30-day fees will be charged as follows:

Upload traffic fees for using an **acceleration endpoint domain name**: $30 \times 1 \text{ GB} \times (0.07 \text{ USD/GB} + 0.18 \text{ USD/GB}) = 7.5 \text{ USD}$

Upload traffic fees for using a **default bucket endpoint domain name**: $30 \times 1 \text{ GB} \times (0 \text{ USD/GB}) = 0 \text{ USD}$

Note:

Upload traffic fees within the Chinese mainland are charged at 0.07 USD/GB, while those outside the Chinese mainland are charged at 0.18 USD/GB when an acceleration endpoint domain name is used. Uploads with a default bucket endpoint domain name do not incur fees.

Business scenario 2

A user uses COS mainly to download video files that require a high transmission success rate. If he/she downloads 1 GB of video data daily from his/her office in Singapore to his/her bucket in Guangzhou, then 30-day fees will be charged as follows:

Download traffic fees for using an acceleration domain name: $30 \times 1 \text{ GB} \times 0.18 \text{ USD/GB} = 5.4 \text{ USD}$

Public downstream traffic fees for using an acceleration domain name: $30 \times 1 \text{ GB} \times 0.1 \text{ USD/GB} = 3 \text{ USD}$

To sum up, the total download traffic fee is 8.4 USD ($5.4 + 3 = 8.4$).

Note:

The unit price for cross-border download acceleration is 0.18 USD/GB. If you use a global acceleration domain name to download files, **public downstream traffic fees** and **global acceleration downstream traffic fees** will be charged.

Private Network Global Acceleration

Last updated : 2024-03-25 15:33:39

By leveraging Tencent Cloud's global traffic scheduling capabilities, COS's private network global acceleration routes and resolves user requests intelligently to select the optimal network access linkage, helping you quickly access resources across regions over the private network. It allows you to fully enjoy the stable transfer quality of Tencent Cloud's private network backbone lines, so as to improve the stability and performance of cross-region data transfer. Many businesses need to pull data across regions over the private network. In the container image distribution scenario, for example, image repositories are usually stored in a region centrally, and the container cluster may be deployed in different regions based on the business needs. During application deployment, the container cluster needs to pull the image from another region, which generates many cross-region requests. In this scenario, a private network global acceleration endpoint can be used to offer a stable private network cross-region Direct Connect line to improve the image distribution stability.

Note:

Using this feature incurs fees as request data transfers are accelerated via Direct Connect lines in the Tencent Cloud private network. For more pricing information, see [Pricing | Cloud Object Storage](#).

How to Use

You can enable global acceleration in the COS console or through APIs.

Using COS console

You can enable global acceleration for your buckets in the COS console. For more information, see [Enabling Global Acceleration](#) in the console documentation.

Using RESTful APIs

You can directly use the following APIs to enable global acceleration:

[PUT Bucket Accelerate](#)

[GET Bucket Accelerate](#)

Access Endpoint Domain Names

After enabling the global acceleration feature, you can access COS files in another region over the private network through a private network global acceleration endpoint in the format of `<BucketName-APPID>.cos-internal.accelerate.tencentcos.cn`.

For example, a business' application image is stored in bucket `examplebucket-1250000000` in Guangzhou region, and its container clusters are deployed in Guangzhou, Beijing, and Shanghai regions. To accelerate container image distribution, the business team enables global acceleration for bucket `examplebucket-1250000000`, so container clusters in Beijing, Shanghai, and Guangzhou regions can pull the image package over the private network through the private network global acceleration endpoint `examplebucket-1250000000.cos-internal.accelerate.tencentcos.cn`.

When the container cluster in Guangzhou region pulls the file through the endpoint, COS will intelligently resolve the request to the private network access layer in Guangzhou region to directly pull the file from the storage cluster in the same region.

When the container cluster in Beijing/Shanghai region pulls the file through the endpoint, COS will intelligently resolve the request to the private network access layer in Beijing/Shanghai region and use the private network access layer device to pull the file from the storage cluster in Guangzhou region over the cross-region backbone network Direct Connect line.

Note:

If you use global acceleration, fees will be incurred. Therefore, we recommend that you carefully evaluate whether to use this feature with your actual business needs in mind:

The private network global acceleration endpoint can be used in the Tencent Cloud private network environment only. If your request source is not in the Tencent Cloud private network environment, it cannot be connected to. In this case, you can consider using the default endpoint or the default global acceleration endpoint.

When you use a private network global acceleration endpoint in another Tencent Cloud product such as CVM, TKE, or SCF, the product must be in a region where COS is available; otherwise, it cannot be used. For more information on regions where COS is available, see [Regions and Access Endpoints](#).

Limits

We have outlined below some important factors to note when using a global acceleration endpoint domain name:

The global acceleration domain name will take effect 15 minutes after being enabled.

After the global acceleration domain name is enabled, the maximum bandwidth for a bucket will be allocated based on the business volume of the entire network.

After the global acceleration domain name is enabled, only requests using that domain name will be accelerated. However, the default bucket domain name can still be used.

When using a global acceleration domain name, fees will be incurred only for requests for which linkage is accelerated. For example, if you use a global acceleration domain name to upload data from Beijing to a bucket in Beijing, the request will not incur acceleration fees as the linkage was not accelerated.

When using a global acceleration domain name, you can specify the HTTP or HTTPS transfer protocol. However, if the request is transmitted via a private network Direct Connect line, COS will choose to use HTTPS protocol to guarantee data transfer security.

Data Workflow

Overview

Last updated : 2024-03-25 15:33:39

Overview

Data workflow is a new data processing service launched by COS. It offers two features: workflow and task.

Workflow: With the workflow feature, you can quickly and flexibly create data processing processes as needed. A workflow is bound to a path of an input bucket. When a file is **uploaded** to the path, the workflow will be **automatically triggered** to perform the specified processing operation, with the processing result automatically saved to the specified path of the output bucket.

Task: For files in a bucket, you can create tasks to perform data processing operations.

Template: When using data workflows, you usually need to set a series of parameters, which can be combined through a template. This **simplifies your operations** and allows you to reuse the configured parameters with no need to enter them repeatedly.

Note:

For audio/video transcoding, audio/video splicing, video frame capturing, and animated image generation, you need to specify a template when creating a task or workflow, which can be either a **preset** or **custom** one.

Queue and callback: When you activate the data workflow service, the system will **automatically create** a user queue for you. When you submit a task, the task will be arranged in the queue first and executed in sequence according to the priority and order of submission. You can also set a **callback rule** to stay up to date with the task or workflow progress, and the system will send the processing result and status information to the specified address.

The processing operations currently supported by data workflow include:

Operation	Description
Audio/Video transcoding	Converts an audio/video file bitstream. It changes parameters of the source bitstream, such as codec, resolution, and bitrate, to adapt to different devices and network conditions.
TESHD transcoding	Provides transcoding capabilities that make videos smaller and clearer. It integrates a complete set of video processing solutions such as image remastering and enhancement, adaptive content parameter selection, and V265 encoder to deliver a better visual experience with guaranteed low network resource usage.
Professional media format transcoding	Can transcode special formats such as XAVC and ProRes.
Video	Optimizes the video image quality and enhances the visual effect through a series of features,

enhancement	including details enhancement, color enhancement, and SDR to HDR conversion.
Super-resolution	Reconstructs the details and local features of a video by recognizing its content and contour so as to generate a high-resolution video image through a series of low-resolution video images. It can be used in combination with video enhancement to remaster old videos.
Highlight generation	Accurately extracts highlights from a video by recognizing and aggregating the video content, postures, and scenes and quickly clipping them professionally.
Voice/Sound separation	Separates the human voice and background sound in a specified video (or audio) to generate audio materials for subsequent artistic processing of other styles.
Adaptive HLS muxing	Generates a multi-bitrate adaptive file from a raw video to adapt the video to different devices and network conditions.
HDR to SDR conversion	Makes the image details of the output video as close as possible to those of the original video to adapt to different types of devices and avoid image distortion and darkness.
Video frame capturing	Captures the frames of a video at specified time points. You can customize the start time point of frame capturing, frame capturing interval, number of frames to be captured, and output image size and format to meet your diversified needs.
Audio/Video splicing	Adds a video/audio segment at the beginning or end of a video/audio file to generate a new one.
Audio/Video segmentation	Divides a video/audio file into several segments of the specified duration.
Animated image generation	Converts a video file into an animated image file. You can specify the video segment for conversion, frame sampling method, as well as the frame rate, size, and format of the output animated image to meet your different needs.
Intelligent thumbnail generation	Intelligently analyzes the quality, brilliance, and content relevance of video frames by understanding the video content with Tencent Cloud's advanced AI technologies. Then, it extracts optimal frames to generate thumbnails to make the content more engaging.
Speech recognition	Recognizes an audio recording file and returns the recognized text asynchronously. Currently, the supported languages include Mandarin, English, and Cantonese. Meanwhile, Cloud Infinite (CI) can help you process the recognition result, such as blocking impolite words, filtering interjections, and intelligently converting numbers to words, meeting your various speech recognition needs.
File preview	Allows you to preview files of nearly 30 types through image or HTML online, with the source file style preserved as much as possible. This addresses the lack of support for certain file formats on different devices and enables online file preview on PC, app, and other terminals.
Image processing	Supports flexible image editing, such as rotation, cropping, transcoding, and zooming. It provides multiple image downsizing solutions like Guetzli compression, TPG transcoding, and

HEIF transcoding, as well as diversified copyright protection solutions like image/text/blind watermarking. This meets your image processing needs in different business scenarios.

Use Cases

Multi-device adaptability

As content platforms are generally intended for multiple types of devices, they need to provide media files in different formats for different users. The audio/video transcoding feature covers most transcoding needs and provides diversified compression capabilities to increase the compression efficiency and downsize files. This reduces lags, storage space usage, and traffic fees.

Video platform

For traditional video platforms, reviewers need to watch videos and then manually select thumbnails, which is labor consuming and slows down video release.

The intelligent thumbnail generation feature can quickly select the most striking frames as thumbnails, which saves labor resources and accelerates video release.

The animated image generation feature allows you to select the highlights in a video to convert them into an animated image for video preview, so that users can get a glimpse of the video without playing it back. Compared with traditional static video thumbnails, animated image thumbnails increase the click rate and video playbacks.

Video subtitles generation

A list of words and corresponding timestamps can be generated for audio files, which makes it easier to subtitle corresponding videos.

Meeting recording transcription

The minutes of large meetings are complex. If a meeting is lengthy and attended by many people, it would be more difficult to keep the complete minutes. The speech recognition feature of COS can recognize Mandarin, English, and Cantonese, which reduces the workload of taking meeting minutes and improves the meeting effect.

Online education

The file preview feature enables viewing various types of files such as courseware and handouts in online education, which delivers an easier user experience.

Website transcoding

The display of file content at websites is subject to browser rules. The file preview feature of COS allows generating images from multiple types of files for preview. This addresses the display problems of file content on webpages.

Directions

Through COS console

You can use the data workflow service in the COS console. For more information, see the documentation for [workflow](#) and [task configuration](#).

Through RESTful APIs

You can use APIs to manage workflows and tasks. For more information, see the data workflow API documentation.

Monitoring and Alarms

Last updated : 2024-03-25 15:33:39

Overview

COS statistics such as read and write requests and traffic are collected and displayed based on [Cloud Monitor \(CM\)](#). You can view detailed monitoring data of COS in the COS or [CM](#) console.

Note:

This document describes how to get statistics in the COS console. You can call CM APIs to get more detailed data. For more information, see the [CM documentation](#).
Currently, all the metrics reported to CM support all COS regions. For more information, see [Regions and Access Endpoints](#).

Basic Features

CM provides the following modules for COS to implement monitoring and alarming.

Module	Capability	Main Feature
Monitoring overview	Displays the current status of the product	Provides general overview, alarm overview, and overall monitoring information
Alarm management	Supports alarm management and configuration	Supports creating new COS alarm policies, custom messages, and trigger templates
Monitoring platform	Monitors traffic and displays data of user-defined monitoring metrics	Displays your overall bandwidth information and allows you to customize monitoring metrics and data to be reported
Cloud product monitoring	Displays the COS bucket monitoring view	Allows you to query the current monitoring views and data such as read/write requests and traffic for each bucket

Use Cases

Daily management: You can log in to the CM console to view the running status of COS in real time.
Troubleshooting: You can receive alarm notifications when the data of a monitoring metric reaches the threshold. It allows you to quickly notify the exceptions, find out the causes, and fix the issues.

Setting and Querying via Console

You can create an alarm policy for COS in the [CM console](#). If the data of a monitoring metric reaches the specified threshold, you will receive an alarm notification. For detailed directions, see [Setting Alarm Policies](#).

You can go to **Cloud Product Monitoring > Cloud Object Storage** to view the COS monitoring data (including the monitoring data of all buckets, health status, number of alarm policies, and more). Alternatively, you can go to the COS console to view the data. For detailed directions, see [Viewing Statistics](#) and [Querying Monitoring Data](#).

Querying Monitoring Data via APIs

You can call the corresponding APIs to view the COS monitoring data. The monitoring metrics are described below. For more information on monitoring APIs, see [COS](#).

Monitoring Metrics

Note:

COS uses a generic region. Therefore, select **Guangzhou** for **Region** when you pull COS monitoring metric data, regardless of where the bucket resides.

When pulling data by using [API Explorer](#), select **ap-guangzhou** for the `Region` field.

When pulling data by using an SDK, enter "ap-guangzhou" for the `Region` field.

Request metrics

Metric	Meaning	Description	Unit	Dimension
StdReadRequests	STANDARD read requests	Number of STANDARD read requests, which is calculated based on the number of requests sent	-	appid, bucket
StdWriteRequests	STANDARD write requests	Number of STANDARD write requests, which is calculated based on the number of requests sent	-	appid, bucket
IaReadRequests	STANDARD_IA read requests	Number of STANDARD_IA read requests, which is calculated based on the number of requests sent	-	appid, bucket
IaWriteRequests	STANDARD_IA write requests	Number of STANDARD_IA write requests, which is calculated based on the number of requests sent	-	appid, bucket

DeepArcReadRequests	DEEP ARCHIVE read requests	Number of DEEP ARCHIVE read requests, which is calculated based on the number of requests sent	-	appid, bucket
DeepArcWriteRequests	DEEP ARCHIVE write requests	Number of DEEP ARCHIVE write requests, which is calculated based on the number of requests sent	-	appid, bucket
ItReadRequests	INTELLIGENT TIERING read requests	Number of INTELLIGENT TIERING read requests, which is calculated based on the number of requests sent	-	appid, bucket
ItWriteRequests	INTELLIGENT TIERING write requests	Number of INTELLIGENT TIERING write requests, which is calculated based on the number of requests sent	-	appid, bucket
TotalRequests	Total requests	Total number of read and write requests in all storage classes, which is calculated based on the number of requests sent	-	appid, bucket
GetRequests	Total GET requests	Total number of GET requests in all storage classes, which is calculated based on the number of requests sent	-	appid, bucket
PutRequests	Total PUT requests	Total number of PUT requests in all storage classes, which is calculated based on the number of requests sent	-	appid, bucket

Storage metrics

Metric	Description	Unit	Dimension
StdStorage	STANDARD - storage capacity	MB	appid, bucket
SiaStorage	STANDARD_IA - storage capacity	MB	appid, bucket
ItFreqStorage	INTELLIGENT_TIERING - frequent storage space	MB	appid, bucket
ItInfreqStorage	INTELLIGENT_TIERING - infrequent storage space	MB	appid, bucket
ArcStorage	ARCHIVE - storage capacity	MB	appid, bucket

DeepArcStorage	DEEP ARCHIVE - storage capacity	MB	appid, bucket
StdObjectNumber	STANDARD - number of objects	-	appid, bucket
IaObjectNumber	STANDARD_IA - number of objects	-	appid, bucket
ItFreqObjectNumber	INTELLIGENT TIERING - number of frequently accessed objects	-	appid, bucket
ItInfreqObjectNumber	INTELLIGENT TIERING - number of infrequently accessed objects	-	appid, bucket
ArcObjectNumber	ARCHIVE - number of objects	-	appid, bucket
DeepArcObjectNumber	DEEP_ARCHIVE - number of objects	-	appid, bucket
StdMultipartNumber	STANDARD - number of incomplete multipart uploads	-	appid, bucket
IaMultipartNumber	STANDARD_IA - number of incomplete multipart uploads	-	appid, bucket
ItFrequentMultipartNumber	INTELLIGENT TIERING (frequent access tier) - number of incomplete multipart uploads	-	appid, bucket
ArcMultipartNumber	ARCHIVE - number of incomplete multipart uploads	-	appid, bucket
DeepArcMultipartNumber	DEEP ARCHIVE - number of incomplete multipart uploads	-	appid, bucket

Traffic metrics

Metric	Meaning	Description	Unit	Dimension
InternetTraffic	Public network downstream traffic	Traffic generated by downloading data from COS to a client over the Internet	Byte	appid, bucket
InternetTrafficUp	Public network upstream traffic	Traffic generated by uploading data from a client to COS over the Internet	Byte	appid, bucket

InternalTraffic	Private network downstream traffic	Traffic generated by downloading data from COS to a client over a Tencent Cloud private network	Byte	appid, bucket
InternalTrafficUp	Private network upstream traffic	Traffic generated by uploading data from a client to COS over a Tencent Cloud private network	Byte	appid, bucket
CdnOriginTraffic	CDN origin-pull traffic	Traffic generated by transferring data from COS to Tencent Cloud CDN edge node	Byte	appid, bucket
InboundTraffic	Total upload traffic over public and private networks	Traffic generated by uploading data from a client to COS over a Tencent Cloud private network or the Internet	Byte	appid, bucket
CrossRegionReplicationTraffic	Cross-region replication traffic	Traffic generated by replicating data to a bucket in another region	Byte	appid, bucket

Return code metrics

Metric	Meaning	Description	Unit	Dimension
2xxResponse	2xx status code	Number of requests with a 2xx status code returned	-	appid, bucket
3xxResponse	3xx status code	Number of requests with a 3xx status code returned	-	appid, bucket
4xxResponse	4xx status code	Number of requests with a 4xx status code returned	-	appid, bucket
5xxResponse	5xx status code	Number of requests with a 5xx status code returned	-	appid, bucket
2xxResponseRate	Proportion of 2xx status code	Proportion of the requests with a 2xx status code returned in the total requests	%	appid, bucket
3xxResponseRate	Proportion of 3xx status code	Proportion of the requests with a 3xx status code returned in the total requests	%	appid, bucket
4xxResponseRate	Proportion of 4xx status code	Proportion of the requests with a 4xx status code returned in the total requests	%	appid, bucket

5xxResponseRate	Proportion of 5xx status code	Proportion of the requests with a 5xx status code returned in the total requests	%	appid, bucket
400Response	400 status code	Number of requests with a 400 status code returned	-	appid, bucket
403Response	403 status code	Number of requests with a 403 status code returned	-	appid, bucket
404Response	404 status code	Number of requests with a 404 status code returned	-	appid, bucket
400ResponseRate	Proportion of 400 status code	Proportion of the requests with a 400 status code returned in the total requests	%	appid, bucket
403ResponseRate	Proportion of 403 status code	Proportion of the requests with a 403 status code returned in the total requests	%	appid, bucket
404ResponseRate	Proportion of 404 status code	Proportion of the requests with a 404 status code returned in the total requests	%	appid, bucket
500ResponseRate	Proportion of 500 status code	Proportion of the requests with a 500 status code returned in the total requests	%	appid, bucket
501ResponseRate	Proportion of 501 status code	Proportion of the requests with a 501 status code returned in the total requests	%	appid, bucket
502ResponseRate	Proportion of 502 status code	Proportion of the requests with a 502 status code returned in the total requests	%	appid, bucket
503ResponseRate	Proportion of 503 status code	Proportion of the requests with a 503 status code returned in the total requests	%	appid, bucket

Note:

1. For more information on 3xx, 4xx, and 5xx status codes, see [Error Codes](#).
2. The statistical granularity (`period`) may vary by metric. You can call the [DescribeBaseMetrics](#) API to obtain the `period` supported by each metric.

Data retrieval metrics

Metric	Meaning	Description	Unit	Dimension
StdRetrieval	STANDARD data retrieval	Traffic generated by the retrieval of STANDARD data, which is the sum of the public network downstream traffic, private network downstream	B	appid, bucket

		traffic, and CDN origin-pull traffic in the STANDARD storage class		
laRetrieval	STANDARD_IA data retrieval	Traffic generated by the retrieval of STANDARD_IA data, which is the sum of the public network downstream traffic, private network downstream traffic, and CDN origin-pull traffic in the STANDARD_IA storage class	B	appid, bucket

Data processing metrics

You can call the corresponding APIs to view the CI monitoring data. For more information on monitoring APIs, see [CI Monitoring Metrics].

Dimensions and Parameters

Parameter	Dimension	Description	Format
&Instances.N.Dimensions.0.Name	appid	Dimension name of the root account APPID	Enter a string-type dimension name: appid
&Instances.N.Dimensions.0.Value	appid	Specific root account APPID	Enter a root account APPID, such as 1250000000
&Instances.N.Dimensions.1.Name	bucket	Dimension name of the bucket	Enter a string-type dimension name: bucket
&Instances.N.Dimensions.1.Value	bucket	Specific bucket name	Enter a specific bucket name, such as examplebucket-1250000000

Input Parameters

To query COS monitoring data, the values of the input parameters are as follows:

&Namespace=QCE/COS

&Instances.N.Dimensions.0.Name=appid

&Instances.N.Dimensions.0.Value=root account APPID

&Instances.N.Dimensions.1.Name=bucket

&Instances.N.Dimensions.1.Value=bucket name

Monitoring Description

Monitoring interval: CM supports multiple monitoring intervals, including real time, last 24 hours, last 7 days, and user-specified period, with time granularities of 1 minute, 5 minutes, 1 hour, and 1 day.

Data storage: 1-minute monitoring data can be stored for 15 days, 5-minute data for 31 days, 1-hour data for 93 days, and 1-day data for 186 days.

Alarm display: CM integrates the monitoring data of COS and displays the data in graphs. Alarm notifications can be sent to you according to the predefined alarm metrics of your product. In this way, you can stay informed of the overall running status.

Alarm settings: You can set the threshold for the monitoring metrics. When the monitoring data meets the alarm condition that is set, CM will send the alarm notifications to the specified users. For more information, see [Alarm Overview](#) and [Setting Alarm Policies](#).