Cloud Object Storage

Product Documentation

Contents

Cloud Object Storage 6
- Product Introduction 6
  - COS Overview 6
  - Features Overview 7
  - Specifications and Limits 10
- Purchase Guide 12
  - Billing Overview 12
  - Purchase Guide 14
  - Instructions on Arrears 15
  - Viewing Consumption Details 16
- Quick Start 17
  - Basic Concepts 17
  - Creating a Bucket 18
  - Uploading an Object 19
  - Downloading Objects 20
  - Deleting an Object 21
  - Deleting a Bucket 22
  - Following Steps 23
- Console Guide 24
  - Console Overview 24
  - Bucket Management 26
    - Creating and Deleting 26
    - Browsing and Querying 28
    - Setting Access Permission 29
    - Setting Anti-hotlinking 31
    - Setting Object Lock 34
    - Setting Origin-Pull 35
    - Setting the Bucket Lifecycle 41
    - Setting Cross-Origin Access 45
    - Setting a Static Website 48
    - Sub-Account Access to Bucket List 51
    - Adding Bucket Policies 53
    - Setting Inventory 56
    - Setting Bucket Tags 59
    - Setting Versioning 61
    - Setting Bucket Replication 62
      - Replication Within the Cloud 62
      - Setting Cross-Cloud Replication 64
    - Setting Bucket Encryption 66
    - Configuring Custom Domain 69
    - Setting Log Management 70
    - Setting Event Notifications 72
  - Object Management 74
    - Uploading an Object 74
    - Downloading Objects 77
    - Copying Objects 79
    - Viewing Object Information 81
    - Searching for Objects 83
    - Sorting or Filtering Objects 84
    - Renaming a File 85
    - Modifying Storage Type 86
    - Setting Access Permissions for an Object 88
    - Setting Object Encryption 90
    - Custom Headers 92
    - Deleting an Object 95
    - Folder Management 97
      - Creating a Folder 97
      - Deleting a Folder 98
    - Setting Object Tags 99
    - Restoring Historical Version Objects 102
  - Data Statistics 104
    - Basic Data Statistics 104
    - Return Code Statistics 105
  - Data Monitoring 106
    - Viewing Data Overview 106
    - Viewing Data Monitoring 109
    - Setting Monitoring Alarms 110
  - Advanced Features 112
    - Supporting IPV6 Access 112
    - Region Allowlist 114
- Developer Guide 115
  - Bucket 115
    - Bucket Overview 115
    - Creating a Bucket 117
    - Deleting a Bucket 118
  - Object 119
    - Object Overview 119
    - Uploading Objects 125
      - Simple Upload 125
      - Multipart Upload 126
      - Uploading Pre-signed Authorized 128
    - Getting Objects 129
      - Simply Getting Objects 129
      - Downloading via Pre-Signed URL 131
    - Listing Object Keys 132
    - Replicating Objects 133
      - Simple Copy 133
      - Multipart Copy 134
    - Deleting an Object 135

- Deleting an Object ... 135
- Data Management ... 136
- Lifecycle Management ... 136
- Lifecycle Overview ... 136
- Lifecycle Configuration Elements ... 138
- Configuring Lifecycle ... 143
- Static Website Hosting ... 144
- Inventory Overview ... 147
- Bucket Tag Overview ... 153
- Object Tag Overview ... 155
- Log Management ... 157
- Logging Overview ... 157
- Restrictions on Log Management ... 161
- Data Disaster Recovery ... 163
- Versioning ... 163
- Versioning Overview ... 163
- Deleting Markers ... 173
- Versioning Configuration ... 175
- Replication Within the Cloud ... 176
- Overview of Replication Within the Cloud ... 176
- Replication Actions Description ... 178
- Configuring Replication Within the Cloud ... 180
- Cross-Cloud Replication ... 181
- Cross-Cloud Replication Overview ... 181
- Cross-Cloud Replication Configuration ... 183
- Data Security ... 184
- Server-Side Encryption Overview ... 184
- Bucket Encryption Overview ... 188
- Access Management ... 190
- Access Policy Language Overview ... 190
- Best Practices ... 195
- Access Control and Permission Management ... 195
- ACL Access Control Practice ... 195
- CAM Practices ... 199
- Authorizing a Sub-account to Access COS ... 202
- Cases Related to Permission Settings ... 210
- User Guide to COS API Authorization Policies ... 219
- Security Guidelines for Temporary Keys Used for Direct Transmission from the Frontend to COS ... 239
- Authorizing a Sub-account Under Another Root Account to Operate Buckets Under the Local Root Account ... 243
- Performance Optimization ... 245
- Request Rate and Performance Optimization ... 245
- Data Migration ... 249
- Migrating Data Within COS ... 249
- Data Disaster Recovery and Backup ... 250
- Disaster Recovery and High Availability Architecture Based on Replication Within the Cloud ... 250
- Domain Name Management Practice ... 254
- Setting Cross-Origin Access ... 254
- Hosting a Static Website ... 262
- Data Security ... 265
- Anti-Hotlinking Practice ... 265
- Data Verification ... 268
- MD5 Verification ... 268
- Accessing COS with the AWS S3 SDK ... 271
- API References ... 277
- Overview ... 277
- Common Request Headers ... 278
- Public Response Headers ... 281
- Error Code ... 283
- Request Signing ... 289
- Operation List ... 298
- Service APIs ... 304
- Querying the Bucket List ... 304
- Bucket APIs ... 309
- Basic Operations ... 309
- Creating a Bucket ... 309
- Querying the Object List ... 312
- Retrieving Buckets and Their Permissions ... 318
- Deleting a Bucket ... 320
- Querying the Object Version ... 323
- Access Control List (ACL) ... 346
- Setting an ACL for a Bucket ... 346
- Querying an ACL of a Bucket ... 352
- Cross-Origin Resource Sharing (CORS) ... 357
- Setting CORS Configuration ... 357
- Querying CORS Configuration ... 361
- Deleting CORS Configuration ... 365
- Lifecycle ... 367
- Setting the Bucket Lifecycle ... 367
- Querying the Bucket Lifecycle Configuration ... 373
- Deleting the Bucket Lifecycle Configuration ... 376
- Bucket Policy ... 378
- Setting a Bucket Policy ... 378
- Querying a Bucket Policy ... 381
- Deleting a Bucket Policy ... 384
- Hotlink Protection (Referer) ... 386
- Setting a Bucket Referer ... 386
- Querying a Bucket Referer ... 390
- Tag ... 393
- Setting Bucket Tags ... 393
- Querying Bucket Tags ... 397
- Deleting Bucket Tags ... 400

- Static Website (Website) 403
  - Setting a Static Website 403
  - Querying a Static Website 411
  - Deleting a Static Website 417
- Inventory 419
  - Setting an Inventory Task 419
  - Creating a One-time Inventory Task 430
  - Querying an Inventory Task 435
  - Querying All Inventory Tasks 443
  - Deleting an Inventory Task 450
- Versioning 453
  - Setting Versioning 453
  - Querying Versioning Information 456
- In-cloud Replication (replication) 459
  - Setting Cross-Region Replication 459
  - Querying Cross-Region Replication 462
  - Deleting In-cloud Replication Rules 466
- Cross-Cloud Replication (CrossCloudReplication) 468
  - Setting Cross-Cloud Replication Rules 468
  - Querying Cross-Cloud Replication Rules 472
  - Deleting Cross-Cloud Replication Rules 474
- Logging 476
  - Setting Log Management 476
  - Querying Log Management 479
- Bucket Encryption 482
  - Setting Bucket Encryption 482
  - Querying Bucket Encryption 485
  - Deleting Bucket Encryption 488
- Object Locking 490
  - Setting Object Lock 490
  - Querying the Object Lock 493
  - Querying the Object Locking Expiration Date 496
- Object APIs 498
  - Basic Operations 498
    - Uploading Objects in Simple Upload Mode 498
    - Setting Object Replication 503
    - Uploading Objects Through Forms 507
    - Downloading an Object 514
    - Querying Metadata of Objects 519
    - Deleting an Object 521
    - Deleting Multiple Objects 525
    - Checking CORS Configuration 532
    - Recovering Archived Objects 535
  - Access Control 538
    - Setting an ACL for an object 538
    - Querying the ACL of an Object 543
  - Object Tag 548
    - Setting Object Tags 548
    - Querying Object Tags 552
    - Deleting Object Tags 555
  - Multipart Upload 558
    - Initializing Multipart Upload 558
    - Uploading Parts 562
    - Copying Parts 565
    - Completing Multipart Upload 570
    - Terminating Multipart Upload 575
    - Querying Multipart Upload 578
    - Querying Uploaded Parts 586
- Data Processing APIs 591
  - Image Processing 591
    - Basic Image Processing 591
      - Adjusting Brightness 591
      - Removing Metadata 594
      - Image Watermark 596
      - Contrast 600
      - Quick Thumbnail Template 603
      - List of Supported Fonts 607
      - Text Watermark 608
      - Rotation 612
      - Format Conversion 616
      - Grayscale Image 619
      - Pipeline Operator 621
      - Scaling 624
      - Cropping 630
      - Quality Transformation 639
      - Sharpening 643
      - Limiting the Image Size 646
      - Gaussian Blurring 648
    - Basic Image Compression (WebP) 651
- FAQs 654
  - General 654
  - Billing 657
  - Upload and Download 659
  - Data Security 662
- Glossary 666
- API Documentation 667
  - cos(cos) 667
    - version(2020-01-07) 667
      - API overview 667
      - CallingMethod 668
        - signaturev1 668

- signaturev3 676
- request 685
- result 686
- PublicParameter 689

• Cross-Cloud Replication 691
- CheckBucketCrossCloudReplicationDestination 691
- GetBucketCrossCloudReplication 692
- PutBucketCrossCloudReplication 693

• Data structure 694
• ErrorCode 698

Product Introduction
COS Overview

COS Introduction

Cloud Object Storage (COS) is a distributed storage service provided by the cloud platform for storing a massive number of files. It enables you to store and view data over the network at any time.

- COS provides scalable, affordable, reliable, and secure data storage services for all users.
- COS provides easy and fast access via a variety of methods, including consoles, APIs, and SDKs, realizing the storage and management of massive data.
- COS provides an intuitive web management interface for operations such as file uploading, downloading, and management.

Object Storage Features

Object storage is a non-hierarchical data storage method in cloud storage. It does not use a directory tree structure, and each independent data (object) unit exists at the same level in the storage pool. Each object has a unique identifying name that can be used for listing and searching. Also, each object can contain metadata. Compared with traditional file storage methods, object storage has the following characteristics:

- Data is stored as separate objects.
- Data is not placed in a directory hierarchy, but exists in a flat address space.
- The application identifies each individual data object by a unique address.
- Designed for access at the application level using an HTTP RESTful API.

Features Overview

Before you use COS, it is recommended that you read COS Basic Concepts to learn some basic concepts required for using COS, such as the definitions of bucket, object, region, and access endpoint.

COS mainly offers the following features:

Feature Category	Feature Overview
Operation	- Bucket operations Support creating, querying, deleting, and emptying buckets. - Object operations - Multiple storage types: According to different access frequencies and disaster recovery levels, COS offers multiple object storage types, including STANDARD and STANDARD_IA. - Objects/folders: Support operations such as uploading, querying, downloading, replicating, and deleting.
Data Management	- Lifecycle COS allows you to set rules for objects to perform automatic deletion on a specific object after a certain period of time (in days). - Static website Configure a bucket to host a static website and access the static website via the bucket's domain name. - Inventory COS allows you to configure an inventory task to regularly (daily or weekly) scan your bucket for specified objects or objects with the same prefix. Each task will output an inventory report, which is stored in the specified bucket as a CSV file. - Bucket tag A bucket tag can be used as an identifier for bucket management, so you can group together and manage buckets easily. You can set, query, and delete tags for a specified bucket. - Log management The log management feature is used to log the access details of a specified source bucket and store these details as log files in a specified bucket for better bucket management. - Event management The event notification feature enables prompt notification for user-concerned resource operations within the bucket. - Object tag The object tagging feature is used 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", an equal sign "=", and a "tagValue", for example, "group = IT". You can set, query, and delete tags for a specified object.

Feature Category	Feature Overview
Cross-region disaster recovery	- Versioning Versioning is used to allow you to store multiple versions of an object in the same bucket. After enabling the versioning feature for a bucket, you can search for, delete, or restore the objects stored in the bucket by version ID. This enables you to recover data that was lost due to accidental deletion or application failure. - Bucket replication It is used to allow you to configure the bucket replication rules to automatically and asynchronously replicate incremental objects in different buckets for data disaster recovery and backup.
Data Security	Hotlink protection COS supports configuring hotlink protection. You can configure a blocklist/ allowlist through the hotlink protection feature in the console to protect your data resources.
CAM	- Cross-origin access With COS, you can set cross-origin access in HTML5 standard to enable access across different origins. For cross-origin accesses, COS can respond to OPTIONS requests and return the specified rules to the browser according to the rules set by the developer. - Bucket policy You can add a policy for a bucket to grant or deny an account or source IP (or IP range) access permission for a COS resource. - Access control You can manage the access permissions for your buckets and objects. After receiving a resource request, COS will check the corresponding ACL to verify if the requester has the required access permission.
Data monitoring	- Monitoring and Alarms Data such as the COS read-write request volume and traffic are calculated and displayed based on Cloud Monitor. You can view the detailed monitoring data such as the COS read-write request volume and traffic in the Cloud Monitor console. - Viewing Data Overview COS supports data monitoring. You can query the amount and trends of data of different storage types by different time periods through the monitoring data window.
Tools	Management Tools COS provides multiple useful tools such as COSBrowser, COSCMD, and COS Migration to help you manage and/or migrate data.
API/SDK	APIs: COS offers a rich set of APIs and introduces API usage and parameters, sample requests, sample responses, and error codes. COS offers multiple languages for development: Android, C, C++, .NET, Go, iOS,

Feature Category	Feature Overview
	Java, JavaScript, Node.js, PHP, Python, and Mini Program SDKs.

Specifications and Limits

Category	Specifications and Limits
Bucket	- Limits: 0.1. Once a bucket is created, its name or region cannot be modified. 0.1. Buckets should have unique names under the same user account and cannot be renamed. 0.1. A bucket name can only contain lowercase letters a–z, digits 0–9, hyphen (-), and their combinations. A bucket name can contain 1–50 characters. - Number of buckets: Each root account can have up to 200 buckets (by default). - Number of objects: Each bucket can have an unlimited number of objects in it.
Object	- Limits: Each object key can have a length of 1–850B. - Upload: 0.1. A single object to be uploaded in the console can be up to 512 GB. 0.1. A single object to be uploaded through an API/SDK can be up to 48.82 TB (50,000 GB). Upload API specifications: (a) Simple upload: A single object can be up to 5 GB. (b) Multipart upload: A single object can be up to 48.82 TB. The part size can be 1 MB–5 GB, but the last part can be smaller than 1 MB. The number of parts can be 1–10,000. - Replication: 0.1. Intra-region and cross-region object replication under a single account are supported. 0.1. Intra-region object replication is free of charge, while cross-region object replication incurs traffic fees. 0.1. Replicate APIs specification: (a) Simple replication: A single object to be replicated can be up to 5 GB. (b) Objects larger than 5 GB should be replicated through multipart replication. A single object to be replicated can be up to 48.82 TB. - Batch deletion: Up to 1,000 objects can be deleted at a time when a batch deletion is initiated through APIs or SDKs.
Access policy	Number of rules: For each root account (the same APPID), the total number of policies associated with the bucket and object ACL and Policy and CAM can be up to 1,000.
SDK types	Twelve types: Android, C, C++, .NET, Go, iOS, Java, JavaScript, Node.js, PHP, Python, and Mini Program SDKs.

Purchase Guide
Billing Overview

Billing Mode

Pay-as-you-go (postpaid)

Pay-as-you-go is the default billing mode for COS and is supported in all regions. According to the user's storage capacity, request, traffic, and other metering items, the user account is billed on a daily or monthly basis.

Billing Item

The cost of COS consists of three parts: storage fee, request fee, and traffic fee.

Billing Item	Billing Item Description	Billing Formula
COS Standard Storage Capacity	The billing is calculated based on the storage capacity, different storage types have different unit prices.	Storage capacity fee = storage capacity unit price x daily storage capacity.
COS Standard Storage Read Request	The request cost is calculated based on the number of requests, and the unit price of requests varies for different storage types.	Request fee = price per ten thousand requests x daily cumulative number of requests / 10000.
COS Standard Storage Write Request	The request cost is calculated based on the number of requests, and the unit price of requests varies for different storage types.	Request fee = price per ten thousand requests x daily cumulative number of requests / 10000.
COS Infrequent Storage Capacity	The billing is calculated based on the storage capacity, different storage types have different unit prices.	Storage capacity fee = storage capacity unit price x daily storage capacity.
COS Infrequent Storage Read Request	The request cost is calculated based on the number of requests, and the unit price of requests varies for different storage types.	Request fee = price per ten thousand requests x daily cumulative number of requests / 10000.

Billing Item	Billing Item Description	Billing Formula
COS Infrequent Storage Write Request	The request cost is calculated based on the number of requests, and the unit price of requests varies for different storage types.	Request fee = price per ten thousand requests x daily cumulative number of requests / 10000.
COS Outbound External Network Traffic	Public network downstream traffic.	Traffic fee = price per GB x daily accumulated traffic.
COS Cross-region Replication Traffic	Cross-region Replication Traffic.	Traffic fee = price per GB x daily accumulated traffic.

Billing Cycle

The billing cycle and billing order of each billing item of COS are as follows:

Billing Item	Billing Cycle	Billing Cycle Description
COS Standard Storage Capacity	day	Settle the fees incurred on the previous day daily and generate the bill.
COS Standard Storage Read Request	day	Settle the fees incurred on the previous day daily and generate the bill.
COS Standard Storage Write Request	day	Settle the fees incurred on the previous day daily and generate the bill.
COS Infrequent Storage Capacity	day	Settle the fees incurred on the previous day daily and generate the bill.
COS Infrequent Storage Read Request	day	Settle the fees incurred on the previous day daily and generate the bill.
COS Infrequent Storage Write Request	day	Settle the fees incurred on the previous day daily and generate the bill.
COS Outbound External Network Traffic	day	Settle the fees incurred on the previous day daily and generate the bill.
COS Cross-region Replication Traffic	day	Settle the fees incurred on the previous day daily and generate the bill.

Purchase Guide

COS supports the pay-as-you-go (postpaid) billing mode.

Pay-as-You-Go (Postpaid) Mode:

Once you start using COS services, you will be billed in the pay-as-you-go billing mode by COS by default, and you do not have to purchase the services proactively.

Instructions on Arrears

Bill Cycle

The following table describes the COS billing items and the corresponding billing (bills) cycles.

Billing Item	Billing Cycle
COS Standard Storage Capacity	day
COS Standard Storage Read Request	day
COS Standard Storage Write Request	day
COS Infrequent Storage Capacity	day
COS Infrequent Storage Read Request	day
COS Infrequent Storage Write Request	day
COS Outbound External Network Traffic	day
COS Cross-region Replication Traffic	day

Arrears and Service Suspension

When your account is in arrears, the COS will stop the service after 24 hours, but your data can be retained.

After receiving the arrears notification, you can go to Billing Management > Funds Management > Account Top-up in time to top up to avoid affecting your business.

If you have any questions about the consumption details, you can go to Billing Management > Bill Management > Bill Details page to review and check your consumption details.

Note:

1. After the service is suspended due to arrears, the storage capacity occupied by your data will continue to incur charges, until the data is destroyed.
2. The data cannot be recovered after destroyed.
3. If you renew the account and the balance is not less than 0, the service will be automatically enabled.
4. If the data stored in the COS is no longer used, delete it in time to avoid further charges.

Viewing Consumption Details

You can view the fee information incurred by your account using the object storage service in the Billing Management of the console.

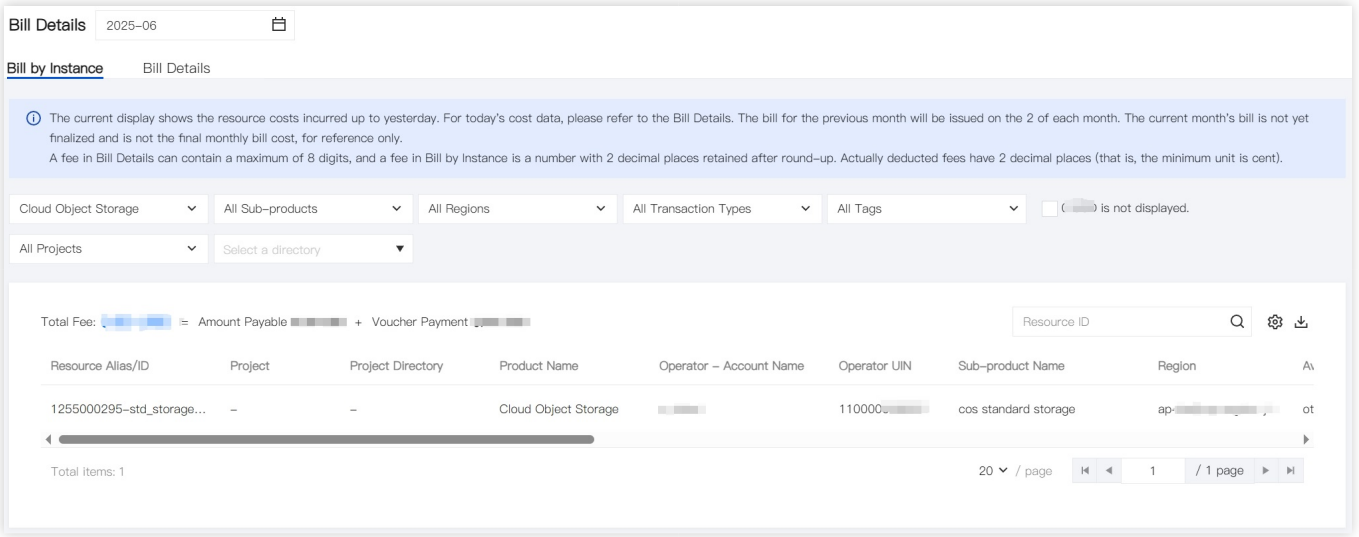
Directions

1. Log in to the Billing Management Console.

2. In the left navigation tree, select Bill Management > Bill Details, and go to Bill by Instance page.

3. Select a COS product to view your COS consumption by region, billing model, transaction type, and other dimensions.
   Check the box not to display 0 USD fees, and the bill center will automatically hide the 0 USD consumption details bill.

4. In the upper right corner of the bill page, click the Download button to download your monthly COS consumption details.



5. Click the Bill Details tab, select COS product, and you can view the detailed bill of COS online.

Quick Start
Basic Concepts

To make the most of COS, you need to learn some basic concepts and terms.

Bucket

A bucket is a container for storing objects in COS. One bucket can store multiple objects. A bucket name is composed of a user-defined string and a system-generated numeric string linked by hyphens to ensure that the bucket is unique across all accounts in all of the cloud platform regions. For more information, see Bucket Overview.

Object

An object is the basic unit stored in COS. For more information, see Object Overview.

APPID

APPID is one of the identifiers for a cloud platform account and is associated with cloud resources. After you successfully sign up for a cloud platform account, you will be automatically assigned an APPID by the system. You can view your APPID at Account Info in the cloud console.

API Key

The API key is the security credential needed for authentication when you access the cloud platform APIs. It consists of SecretId and SecretKey. Multiple Cloud API keys can be created under one user account. If you do not have a Cloud API key, you need to create one in the Cloud API key console. Otherwise, you cannot call any Cloud APIs.

SecretId

As part of the Cloud API Key, SecretId is used to identify the API caller.

SecretKey

As part of the Cloud API Key, SecretKey is the key used to encrypt the signature string and verify the signature string on the server side.

Default Access Address

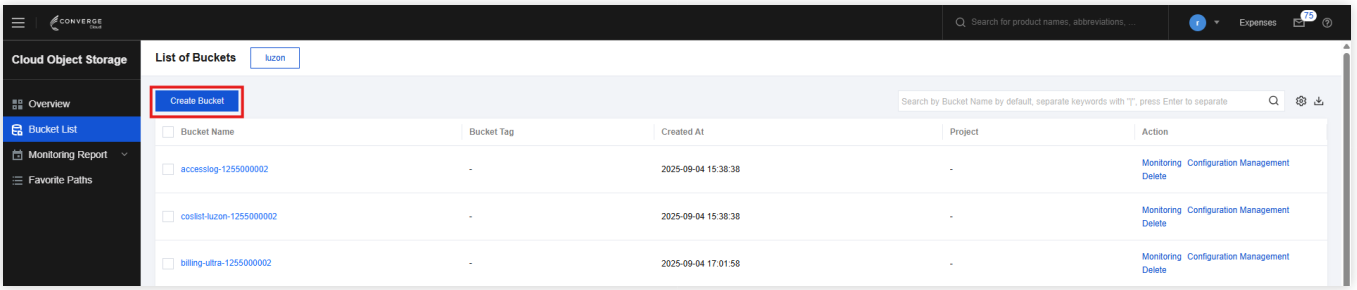
The default access address consists of the bucket name, COS region identifier, and object name. You can address the unique corresponding object in COS via the default access address. After you upload an object, the cloud platform will automatically create a default access address for the object.

Creating a Bucket

For instructions on buckets, see Basic Concepts. You can log in to the COS console and create buckets on the bucket list page:

Directions

1. Log in to the COS console.
2. In the left navigation tree, click Bucket List, and go to the bucket list page.
3. Click Create Bucket.



4. Fill in the bucket name, region, and other information and click OK.
   We recommend that you select an available region that is close to your users. The default access permission is Private Read/Write.

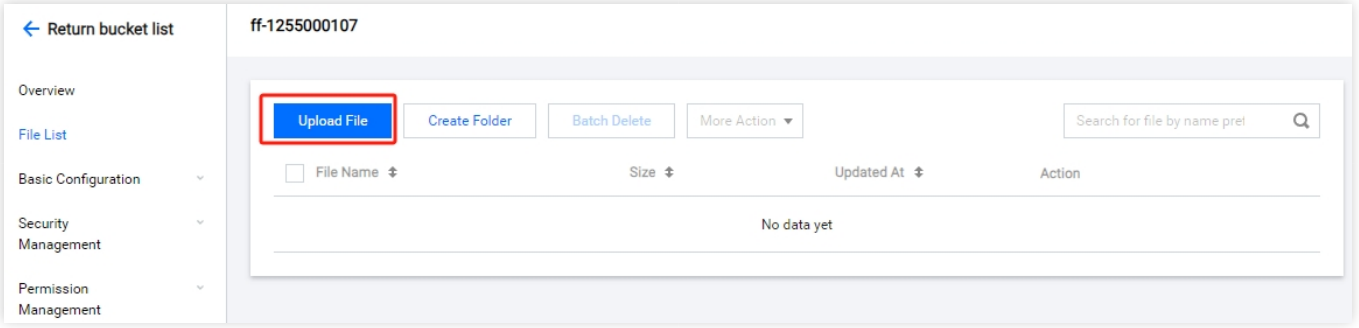
Uploading an Object

After the bucket is created successfully, you can upload objects to the bucket.

Upload Steps

1. Entering the file list

1. Log in to the COS console.
2. In the left navigation tree, click Bucket List to enter.
3. Select the bucket where you want to store objects and go to the file list page.
4. On the File List page, click Upload File.



2. Selecting the upload object

1. In the Upload File dialog box, click Select File or Select Folder to upload single or multiple local files/ folders.
2. Select the local object to be uploaded.
3. Click Upload to start quickly uploading files or click Next to set object attributes.

3. Setting object attributes (optional)

Set the access permissions and metadata information (optional) of the file to be uploaded and click Upload. The access permissions are inherited permissions by default, and metadata information can be left blank for now. After the object is uploaded successfully, the system will automatically refresh the list to obtain the latest object information.

> Note:
>
> Some browsers do not support multiple file uploads. We recommend using mainstream browsers such as IE10 or above, Firefox, and Chrome.

Downloading Objects

Use the access address to download or access objects that are already uploaded to the bucket.

I. Viewing Object Information

1. Log in to the COS console, select Bucket List, and click the corresponding bucket (such as example-1253833564) to enter the file list of the bucket.
2. Find the object to be downloaded (such as example.exe) in the file list, click Details to enter the file information details page.

II. Getting the Object Link and Download

View the file link on the file information details page. You can click the download icon to download the object directly; or click the Copy button to copy the link, and paste it into the browser address bar to access and download the object.
If the attribute of the bucket where the object is stored is private read/write, a signature will be automatically calculated and added as a suffix at the end of the copied address.

Deleting an Object

If you need to delete objects in a bucket, see the following steps:

1. Log in to the COS console.
2. Select Bucket List in the left sidebar to go to the bucket list page.
3. Click the bucket where the object to be deleted is located to enter the bucket file list.
4. Select the object you want to delete and click More Actions > Delete.
5. The delete file dialog box pops up, click Confirm to delete the object.

Deleting a Bucket

You can delete a bucket when you no longer need to use it.

1. On the COS console, you can find the bucket you want to delete and click Delete to pop up the delete bucket dialog box.
2. Click OK to delete the bucket.

> Note:
>
> When deleting a bucket, make sure there are no objects or directories in it. Otherwise, the bucket cannot be deleted.

Following Steps

The previous steps show you how to execute some basic COS tasks in the cloud platform's COS console. The COS console allows you to directly execute most basic tasks. In addition, you can complete more basic and advanced COS tasks by means such as calling the RESTful API and using SDK toolkits for mainstream languages.

Console Guide
Console Overview

Overview

The COS console is one of the tools that help you use COS. You can use the COS console to perform operations such as bucket management and object management without having to write code or run programs. The COS console provides the following features, and their corresponding operation guides are shown in the table below:

Console Menu	Function
Basic Bucket Operations	- Create Bucket - Delete Bucket - Query Bucket - Set Object Lock
Basic Object Operations	- Upload Object - Download Object - Replicate Object - View Object Information - Search for Objects - Sort or Filter Objects - Rename File - Modify Storage Class - Customize Headers - Delete Object - Restore Historical Version of Object
Folder Operations	- Create Folder - Delete Folder
Lifecycle	Set Lifecycle
Tag Management	- Set Bucket Tag - Set Object Tag
Log Management	Set Log Management
Event Notification	Set Event Notification
Versioning	Set Version Control
Cross-Region Replication	Set Cross-Region Replication

Console Menu	Function
Hotlink Protection	Set Hotlink Protection
Cross-Origin Access	Set Cross-Domain Access
Domain Management	Customize Domain Name
Bucket Policy	Add Bucket Policy
Access Control	- Set Bucket Access Permission - Set Object Access Permissions
Monitoring Reports	- Basic Data Statistics - Return Code Statistics

Bucket Management
Creating and Deleting

Directions

Description of Related Information

Name

- The bucket name consists of a custom string and a system-generated numeric string (APPID), which are concatenated by a strike - . The system-generated numeric string (APPID) for different buckets is the same for the same user.
- The length of a custom string in the bucket name cannot exceed 50 characters. Only lowercase letters, digits, and hyphens are allowed. Special symbols and underlines are not allowed.
- Bucket names created by the same user are unique and cannot be renamed.

Access Permission

The bucket provides three access permissions: private read/write, public read and private write, or public read/write. Bucket permissions can be modified through bucket permission management in the COS console.

- Private read/write: Only the creator of the bucket and authorized accounts have read/write permission on the objects in the bucket. No one else has read/write permissions. It is recommended.
- Public read and private write: Anyone (including anonymous visitors) has read permission to objects in the bucket, but only the bucket creator and authorized accounts have write permission to objects in the bucket.
- Public read and write: anyone (including anonymous visitors) has read/write permission on the objects in the bucket, which is not recommended.

Associated Project

- You can select the Project field in the creation process. The project can help you solve the complexity of resource grouping and authorization management in a single cloud account.
- You can go to Resource Management > Projects to create a project.

Creating a Bucket

You can create a bucket on the bucket list page in the COS console.

1. Going to the Bucket List

Log in to the COS console and click Bucket List in the left sidebar to enter the bucket list page. Click Create Bucket to pop up the create bucket dialog box.

2. Filling in the Bucket Information

Fill in the bucket name (such as test-1253833564), create the region where the bucket belongs, set the default private read/write, and click OK to quickly create a bucket.

Deleting a Bucket

You can delete a bucket when you no longer need to use it.

1. On the COS console, find the bucket you want to delete and click Delete.
2. The delete bucket dialog box is displayed. Click OK to delete the bucket.

> Note:
>
> When deleting a bucket, make sure there are no objects or directories in it. Otherwise, the bucket cannot be deleted.

Browsing and Querying

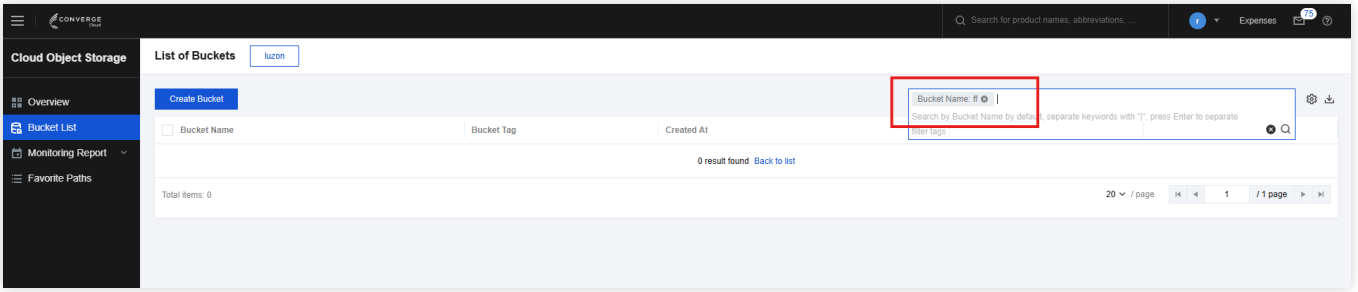
Browsing Buckets

When you need to browse your buckets, in the COS console, click Bucket List on the left menu bar to enter the bucket list page and browse all buckets. If you need more detailed information about a bucket, you can directly click the bucket name to view the bucket.

Querying Buckets

When there are many buckets, and you need to quickly find a specific bucket, you can use the search box on the right side of the bucket list page of the console:

- Query by bucket name: You can enter the bucket name to query, and prefix matching query for bucket names is supported.



- Query by tag: If you have set tags for your bucket (see Setting Bucket Tags for details), you can also select query by tag in the search area in the upper right corner of the bucket list page and enter the tag key to search.
- Query by project: If you have set a project for the bucket, you can also select the project in the search area in the upper right corner of the bucket list page and enter the project to search.

Bucket Monitoring

If you want to know the usage details of your bucket, click Monitoring on the right for a quick view.

Bucket List Exporting

Go to the bucket list page and click Download in the upper right corner to export all bucket information.

Setting Access Permission

Overview

Users can modify bucket access permissions through the console and API. The COS console supports three types of access permissions by default: private read/write, public read and private write, and public read/write.

- Private read/write: Only the creator of the bucket and authorized accounts have read/write permission on the objects in the bucket. No one else has read/write permissions. The default access permission of a bucket is private read/write, which is recommended.
- Public read and private write: Anyone (including anonymous visitors) has read permission to objects in the bucket, but only the bucket creator and authorized accounts have write permission to objects in the bucket.
- Public read and write: anyone (including anonymous visitors) has read/write permission on the objects in the bucket, which is not recommended.

Directions

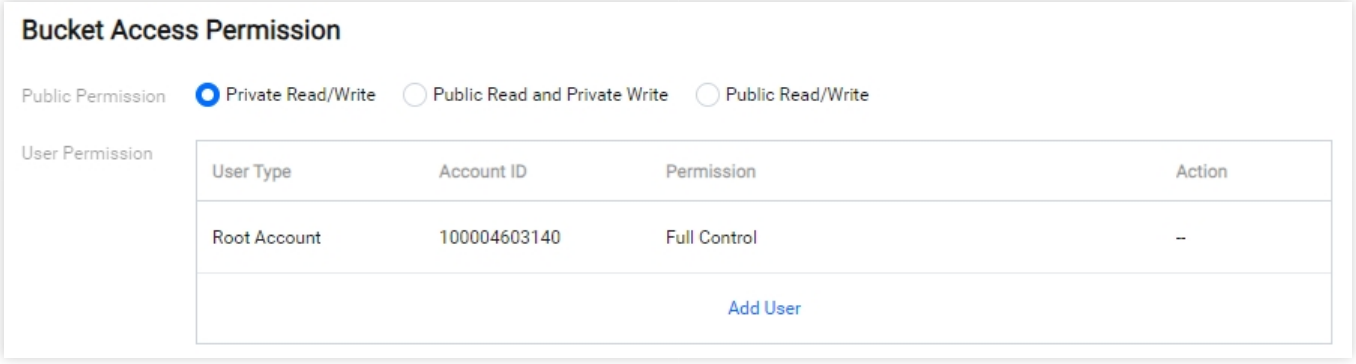
You can set the permission of the bucket when creating a bucket. In addition, you can modify the bucket access permissions through the bucket basic configuration. The specific steps are as follows:

1. Log in to the COS console.

2. In the left navigation tree, click Bucket List, and go to the bucket list page.

3. Click the name of the bucket whose access permissions you want to modify to go to the bucket configuration page.

4. Click Permission Management > Bucket Access Permission to set public permissions and user permissions for the bucket.



5. Click Save after modification.

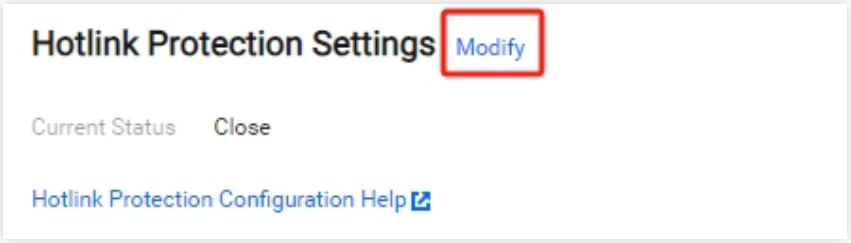
Setting Anti-hotlinking

Overview

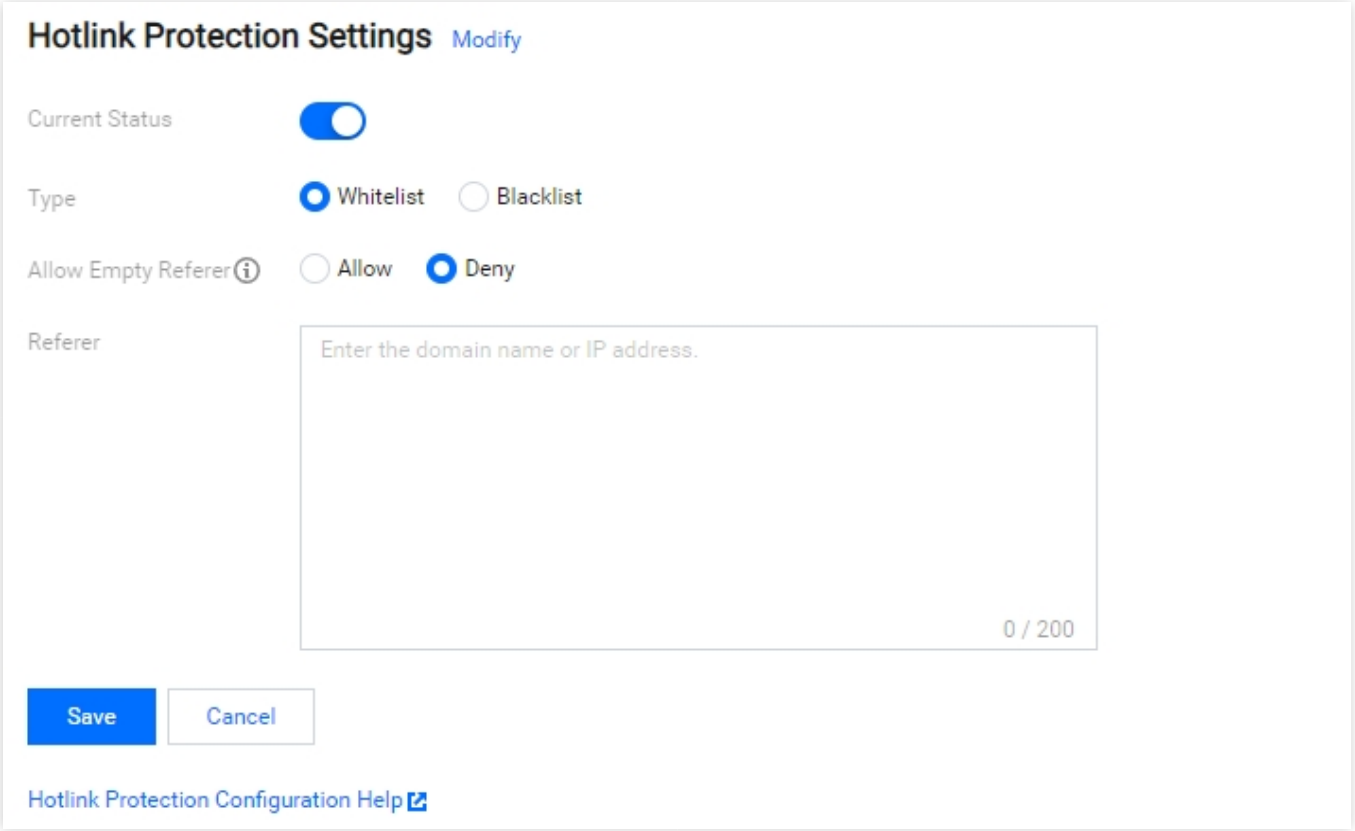
In order to prevent malicious programs from using resource URLs to steal public network traffic or using malicious methods to steal resources, causing unnecessary losses to users. COS supports anti-hotlinking configuration. It is recommended that you configure blocklist/allowlist through the anti-hotlinking settings on the console for security protection.

Directions

1. Log in to the COS console.
2. In the left sidebar, select Bucket List to go to the bucket list page.
3. Locate the bucket for which you want to set anti-hotlinking, and click its name to enter the bucket management page.
4. Select Security Management > Hotlink Protection Settings and click Modify to enter the editing status.



5. Switch the current status to Enable, select a list type (blocklist or allowlist), set the corresponding domain, and then click Save. The configuration items are described as follows:



- Blocklist: Restricts domain names on the list to accessing the default access address of the bucket. If a domain name in the list accesses the default access address of the bucket, a 403 error will be returned.
- Allowlist: Restricts domain names not on the list to accessing the default access address of the bucket. If a domain name not in the list accesses the default access address of the bucket, a 403 error will be returned.
- Allow Empty Referer: The referer in an HTTP request header is left empty (there is no referer field or the referer field is empty).
- Referer: Enter up to 10 domain names or IP addresses (one per line). The wildcards * is supported. For example:
  - Configure www.example.com : It can limits address with the prefix of www.example.com such as www.example.com/123 and www.example.com.cn .
  - Domain names and IPs with ports are supported, such as www.example.com:8080 and 10.10.10.10:8080 .
  - Configure *.example.com : It can limits address such as a.b.example.com/123 and a.example.com .

Example

A user with the APPID of 1250000000 creates a bucket named examplebucket-1250000000 and places an image picture.jpg in the root directory. COS has generated the following default access address according to the rules:

examplebucket-1250000000.cos.city.yfm4.fsphere.cn/picture.jpg

User A owns a website:

www.example.com

And embeds the image into the home page index.html.

Webmaster B manages a website:

www.fake.com

Webmaster B wants to put this picture on www.fake.com . But he does not want to pay for traffic costs. He creates a direct link to picture.jpg through the following address and places it into the homepage index.html on www.fake.com .

examplebucket-1250000000.cos.city.yfm4.fsphere.cn/picture.jpg

In such cases, to avoid losses for User A, we provide the following two methods of anti-hotlinking.

Method One

Configure the Blocklist, fill in *.fake.com to set the domain name, and save it to take effect.

Method Two

Configure the Allowlist, fill in *.example.com to set the domain name, and save it to take effect.

Before Enabled

The image is displayed normally when http://www.example.com/index.html is accessed.
The image is also displayed normally when http://www.fake.com/index.html is accessed.

After Enabled

The image is displayed normally when http://www.example.com/index.html is accessed.
The image cannot be displayed when http://www.fake.com/index.html is accessed.

Setting Object Lock

Overview

You can enable the object locking feature via the console to ensure objects cannot be overwritten or deleted during the set period.

Operation Steps

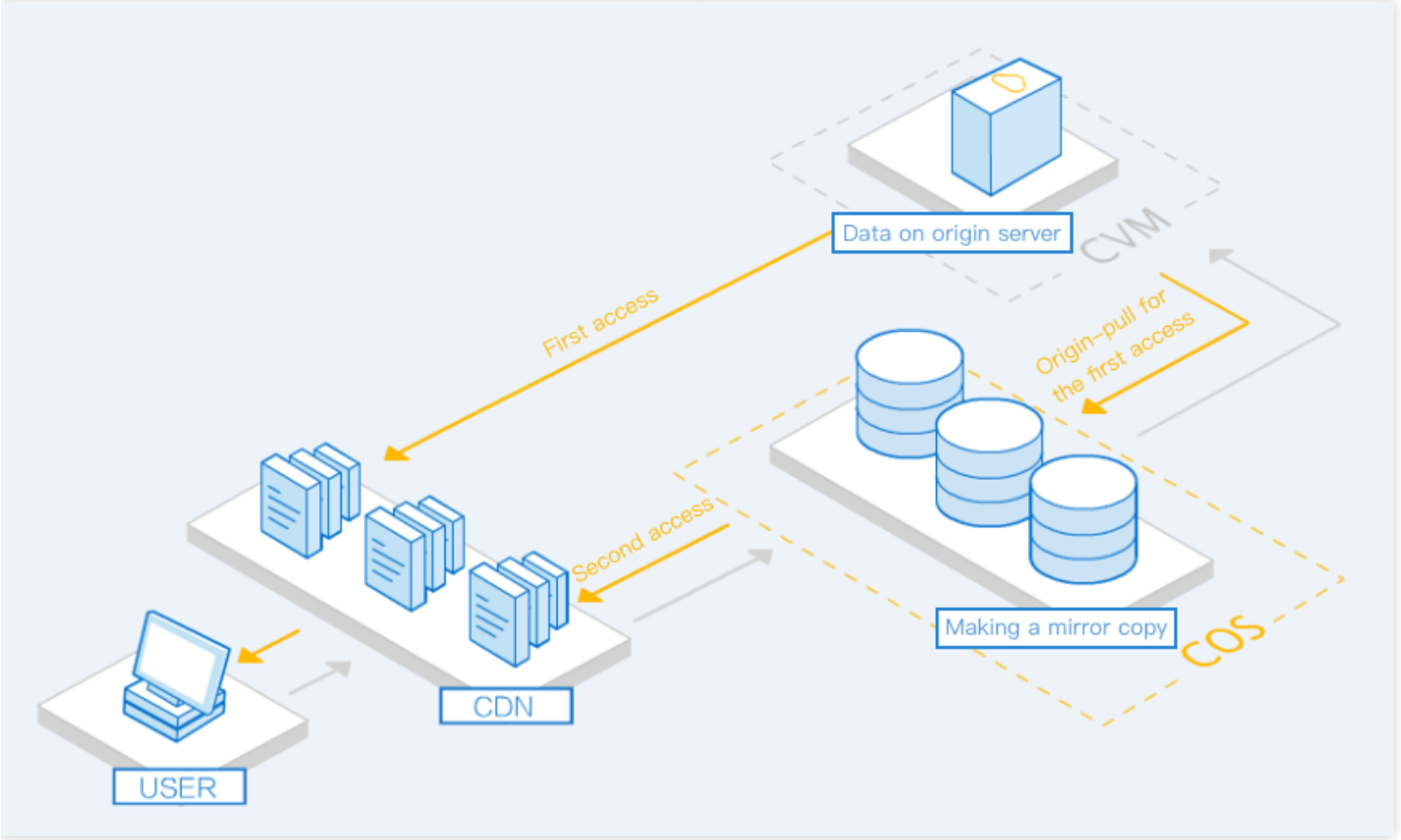
1. Log in to the COS console.
2. In the left sidebar, click Bucket List, and go to the bucket list page.
3. Click the bucket for which you want to set an object lock to enter the file list page.
4. In the left sidebar, choose Security Management > Object Locking and click Modify to change the current status to Enabled.
5. Enter the lock duration in the configuration window and click Save.

Setting Origin-Pull

Overview

You can set origin-pull rules for buckets in the COS console. When the object you request does not exist in the bucket or a specific request needs to be redirected, you can use the origin-pull rules to access the corresponding data from COS. Setting the origin-pull is mainly used for scenes such as hot migration of data and redirection of specific requests. You can set it according to your actual needs.

> Description
>
> - Buckets in the Financial Cloud region do not support origin-pull settings.



Origin-Pull Rules

Trigger Condition

- In asynchronous and synchronous origin-pull modes, origin-pull is triggered only when the request returns a 404 error.

- In redirection mode, users can customize HTTP status codes from 400 to 599 to trigger origin-pull.

Accessing Origin Server

- In asynchronous and synchronous origin-pull modes, you can set whether to pass through the QueryString and Header information of accessing COS to the origin server and whether to carry additional Header information when adding a request to the origin server. The redirection mode only supports setting whether the QueryString is passed through.
- If the GET range is specified in the GET operation, COS will send an asynchronous request without range in addition to the original request to obtain the complete object data and store it in COS.

Response and Storage

- Support the origin server in returning data in chunked encode.

- If the origin server returns a 404 status code, it will be passed through to COS and returned to the user. If 3XX Following Policy is enabled when the origin server returns a 3XX status code, it will pull data from another source server. If the origin server returns a status code other than 2XX, COS will return 424.

- The files returned from the origin-pull are stored in COS with the same file name as used when requesting the origin server. For example, if the file example.jpg requested by a user is not in the storage bucket, COS will trigger the origin-pull mechanism to the origin-pull address http://origin.com/example.jpg set by the user to pull the file and name it example.jpg in the bucket.

- New objects stored in COS contain the following metadata, with the data content following the origin server values:

  cache-control
  content-disposition
  content-encoding
  content-type
  expires
  x-cos-meta-*

Directions

1. Log in to the COS bucket console.

2. In the left sidebar, click Bucket List to enter the bucket list page.

3. Click the name of the bucket for which you want to set origin-pull to go to the bucket details page.

4. In the left sidebar, select Basic Configuration > Origin-pull Settings and click Add an Origin-Pull Rule.

5. Configure the following information in the pop-up.

    - Origin-pull Endpoint: Supports filling in domain or IP address. The correct example address is as follows:

abc.example.com
abc.example.com:8080
202.96.128.86
202.96.128.86:8080

    - Backup origin-pull Endpoint: Supports setting the backup origin-pull address. The input format can refer to the configuration items description of Origin-pull Endpoint.

    - Origin-pull mode: You can choose the origin-pull mode according to your actual needs.
        - Synchronous origin-pull: When a request is made to find a file in COS, and it does not exist, COS will search for the file from the specified origin server, return the file to the user, and upload it to the bucket.
        - Redirection: When a specified error is reported when requesting access to a bucket, COS will return the redirection address to the user and will not save the origin server's files. The user side requests access to resources from the origin server through the redirection address.

    - Origin-pull conditions: You can select the conditions to trigger the origin-pull as required. Triggering the origin-pull requires all configured origin-pull conditions to be met at the same time.
        - HTTP status code 404: When asynchronous or synchronous origin-pull is selected, origin-pull is triggered when the HTTP status code is 404. This option is required and cannot be canceled. When you select the redirection origin-pull mode, you can enter an HTTP status code from 400 to 599.
        - File name prefix: When the requested file name prefix matches, the origin-pull rules can be triggered. For example, if you set the file name prefix to prefix, then visit https://examplebucket-1250000000.cos.city.yfm4.fsphere.cn/prefix123.jpg , and when the

returned HTTP status code is 404, the origin-pull rules are triggered.

- Origin-pull protocol: The HTTP protocol used by COS to access the origin server you specify. The options are Force HTTPS, Force HTTP, and Follow Request Protocol.
  - If you select Force HTTPS/HTTP, COS will access your origin server using the HTTPS/HTTP protocol.
  - If you select Follow Request Protocol, COS will access your origin server using the protocol you requested COS to use.

- Request parameter: Whether to pass through the queryString request parameters carried when accessing COS to the origin server.

- Request header: You can choose to pass through all request headers or pass through specified request headers. If you choose to pass through all request headers, it is recommended that you configure it to prohibit pass-through of the host header. One-click configuration is supported here.

- Pass through specified request headers: When selecting pass through specified request header, you need to set this item. Here, you can add the header information that you want to be passed through to the origin server. If you select the redirection origin-pull type, do not set this item.

- Prohibit pass through specified request headers: Here you can add header information in the request that you do not want to be passed through to the origin server. If you select the redirection origin-pull type, do not set this item.

- Add request headers: You can add additional request headers when origin-pull to the origin server. If you select the redirection origin-pull type, do not set this item.

- Origin-pull path: You can choose whether to set a specific file or file path to jump to. The configuration items are described as follows:
  - Specify file: When the origin-pull rule is triggered, all will jump to a fixed file.
  - Specify prefix/specify suffix: When the origin-pull rule is triggered, it will be redirected to the file path with the specified prefix and suffix. For example, if test is specified as the prefix, visiting https://examplebucket-1250000000.cos.city.yfm4.fsphere.cn/path/prefix123.jpg will trigger the origin-pull rule, and the page will be redirected to <origin-pull address>/test/path/prefix123.jpg ; For example, if .jpg is specified as the suffix, visiting https://examplebucket-1250000000.cos.city.yfm4.fsphere.cn/path/prefix123 will

trigger the origin-pull rule, and the page will be redirected to <origin-pull address>/path/prefix123.jpg ;

> Note
>
> The configurations of specifying a prefix and specifying a suffix can take effect at the same time.

- 3xx following policy: Synchronous origin-pull mode is visible. After it is enabled, when the origin server returns a 3XX redirection status code, COS's default processing policy is to follow 3XX and pull data from another source server again. If you select Disable, no resources will be pulled.
- Redirect Code: Redirection mode is visible. 301, 302, 307 are optional.
- Rule priority: The order of rule execution is determined by priority. Higher priority rules are matched first. Priorities must be unique.

6. Confirm that the configured origin-pull rules are correct and click OK.
   After you add the rules, the system will assign the highest priority to your new rule; COS will origin-pull according to the highest priority rule; in addition, you can also click the modify button on the rule list page to adjust the priority.

Example

Background
The user with APPID 1250000000 creates a bucket named examplebucket-1250000000 and enables CDN to accelerate access to the domain:

```
examplebucket-1250000000.cos.city.yfm4.fsphere.cn
```

Set the bucket origin-pull address to:

```
abc.example.com
```

Store the picture picture.jpg at the origin server http://abc.example.com .
Client's first visit (with synchronous origin-pull disabled):

```
http://examplebucket-1250000000.cos.city.yfm4.fsphere.cn/picture.jpg
```

When COS finds that it cannot hit the object, it returns a 302 HTTP status code to the client and redirects to the following address:

```
http://abc.example.com/picture.jpg
```

Client's first visit (with synchronous origin-pull enabled):

http://examplebucket-1250000000.cos.city.yfm4.fsphere.cn/picture.jpg

When COS finds that it cannot hit the object, it returns a 200 HTTP status code to the client and redirects to the following address:

http://abc.example.com/picture.jpg

At this point, the object is provided to the client by the origin server to ensure access. At the same time, COS copies picture.jpg from the origin server and saves it to the root directory of the bucket example.
Second visit:

http://examplebucket-1250000000.cos.city.yfm4.fsphere.cn/picture.jpg

COS directly hits the picture.jpg object in the root directory and returns it to the client.

Setting the Bucket Lifecycle

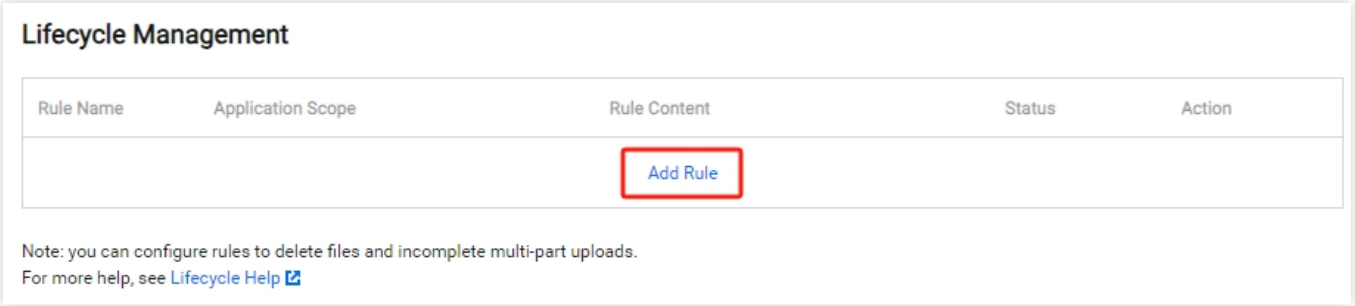
Overview

When you need to periodically delete specified objects to reduce costs, you can use the lifecycle management feature. COS will automatically delete the specified objects within the specified time according to the rules you set.

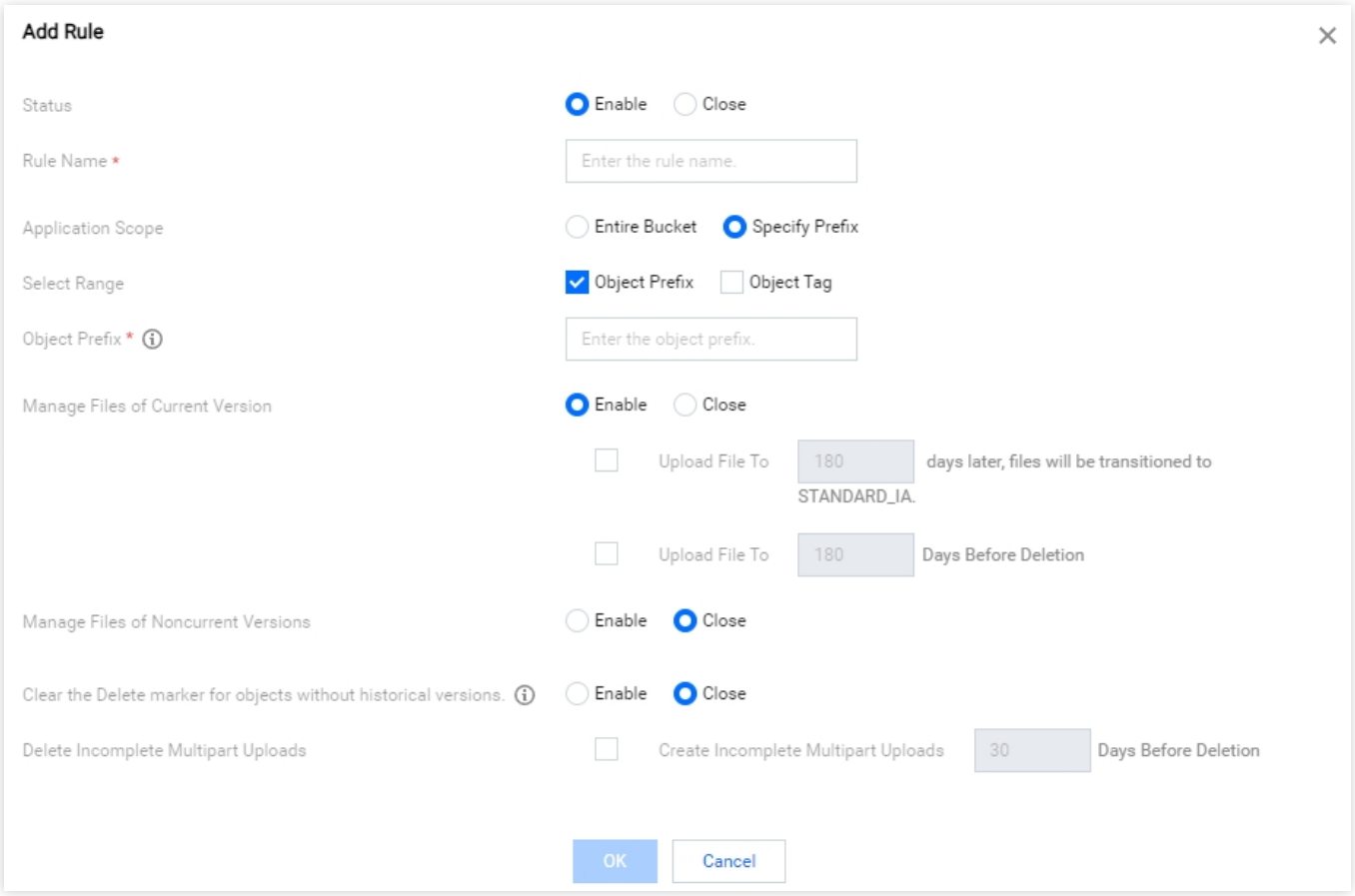
> Note:
>
> A lifecycle can be set to as long as 3,650 days.

Directions

1. Log in to the COS console.
2. In the left sidebar, click Bucket List to enter.
3. Find the bucket for which you want to enable the lifecycle function, click its bucket name, and go to the bucket file list page.
4. Select Basic Configuration > Lifecycle Management and click Add Rule.



5. Add lifecycle rules based on your needs.



The configuration items are described as follows:

- Rule Name : Enter a name for your lifecycle rule.
- Application Scope: This lifecycle rule can be applied to the entire bucket or objects in a specified range. The following ranges are currently supported:

  > Note:
  >
  > Both an object prefix and an object tag can be specified.
  >
  > For information about object keys (or prefixes), see Object Key in the object overview. For a description of the lifecycle configuration rules, see Rule Description documentation.

  - Object Prefix: You can specify objects with the same file prefix to execute lifecycle rules, such as prefix/.
  - Object Tag: You can specify objects with the same tag to execute lifecycle rules. You can specify multiple tags. Distinguish between uppercase and lowercase letters.

- Manage Files of Current Version: You can set file settlement or delete the current version object by enabling the option to manage the current version object. Support objects in the bucket to be settled from standard storage to infrequent access storage and support the deletion of objects after

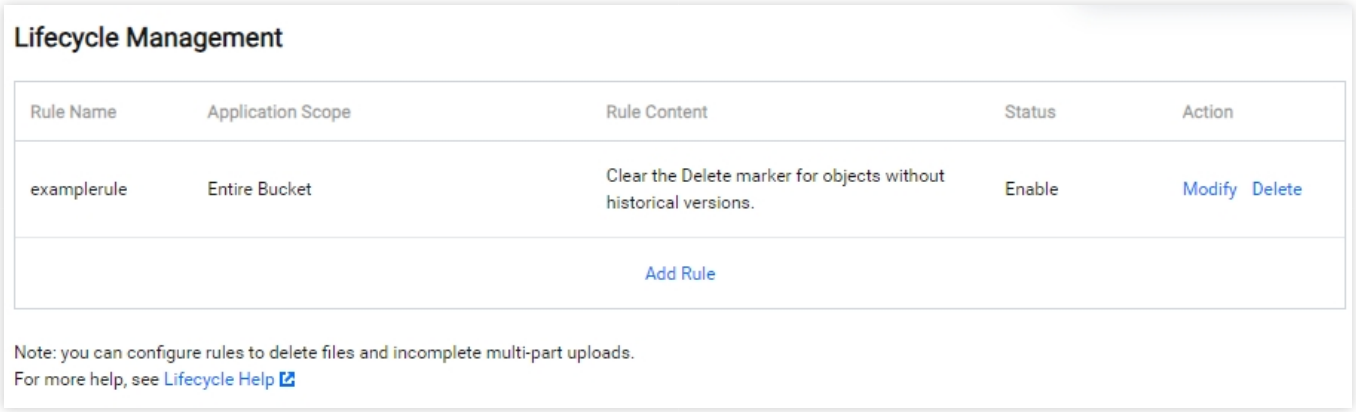
expiration. The time is calculated based on the modification time of the file in the object storage. Modifying an object is equivalent to re-uploading the object.

- Manage Files of Noncurrent Versions: Set it to transition files or delete objects of historical versions by enabling the option to manage objects of historical versions. If this option is disabled, we will only process the latest version of the file by default.

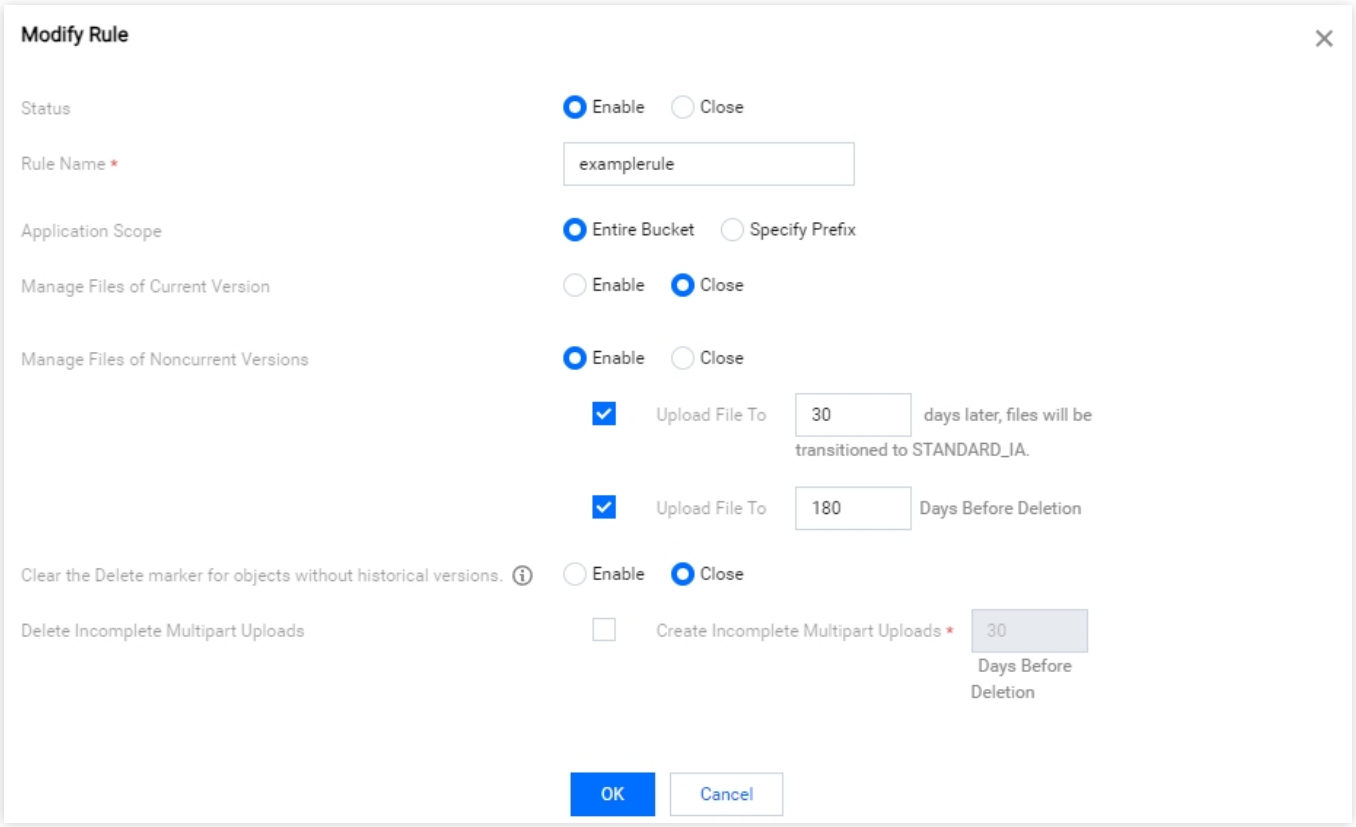
- Clear the Delete marker for objects without historical versions: If the latest version of an object is a delete marker and all historical versions of the object are deleted, this delete marker will be deleted after this option is enabled. This option cannot be enabled at the same time as deletion after expiration in Manage the Current Version File.

- Delete Incomplete Multipart Uploads: When uploading a file, the upload fails due to various reasons, and only a part of it is transmitted. For such damaged files, you can set periodic deletion.

6. After configuring the lifecycle rules, click OK to see the lifecycle rules.



7. If you need to stop a lifecycle rule, click Modify, and change the status of the corresponding rule to Close or click Delete to directly delete the lifecycle rule.


Modify Rule
Status
Enable
Close
Rule Name *
examplerule
Application Scope
Entire Bucket
Specify Prefix
Manage Files of Current Version
Enable
Close
Manage Files of Noncurrent Versions
Enable
Close
Upload File To
30
days later, files will be transitioned to STANDARD_IA.
Upload File To
180
Days Before Deletion
Clear the Delete marker for objects without historical versions.
Enable
Close
Delete Incomplete Multipart Uploads
Create Incomplete Multipart Uploads *
30
Days Before Deletion
OK
Cancel

Setting Cross-Origin Access

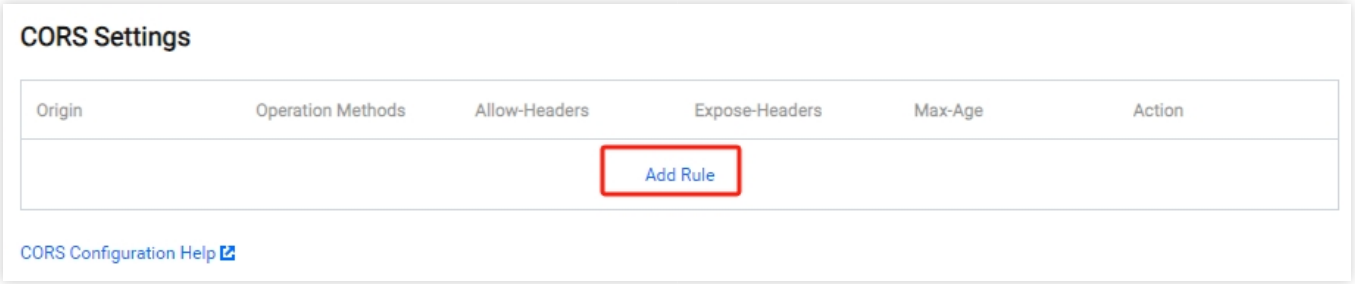
Overview

You can set a cross-origin access for the object in a bucket through the COS console. The COS provides configuration for responding to OPTIONS requests and supports multiple rules. Cross-origin access is to request resources from one origin to another origin through HTTP requests. Origins are deemed different from each other as long as their protocols, domain names, or ports are different.

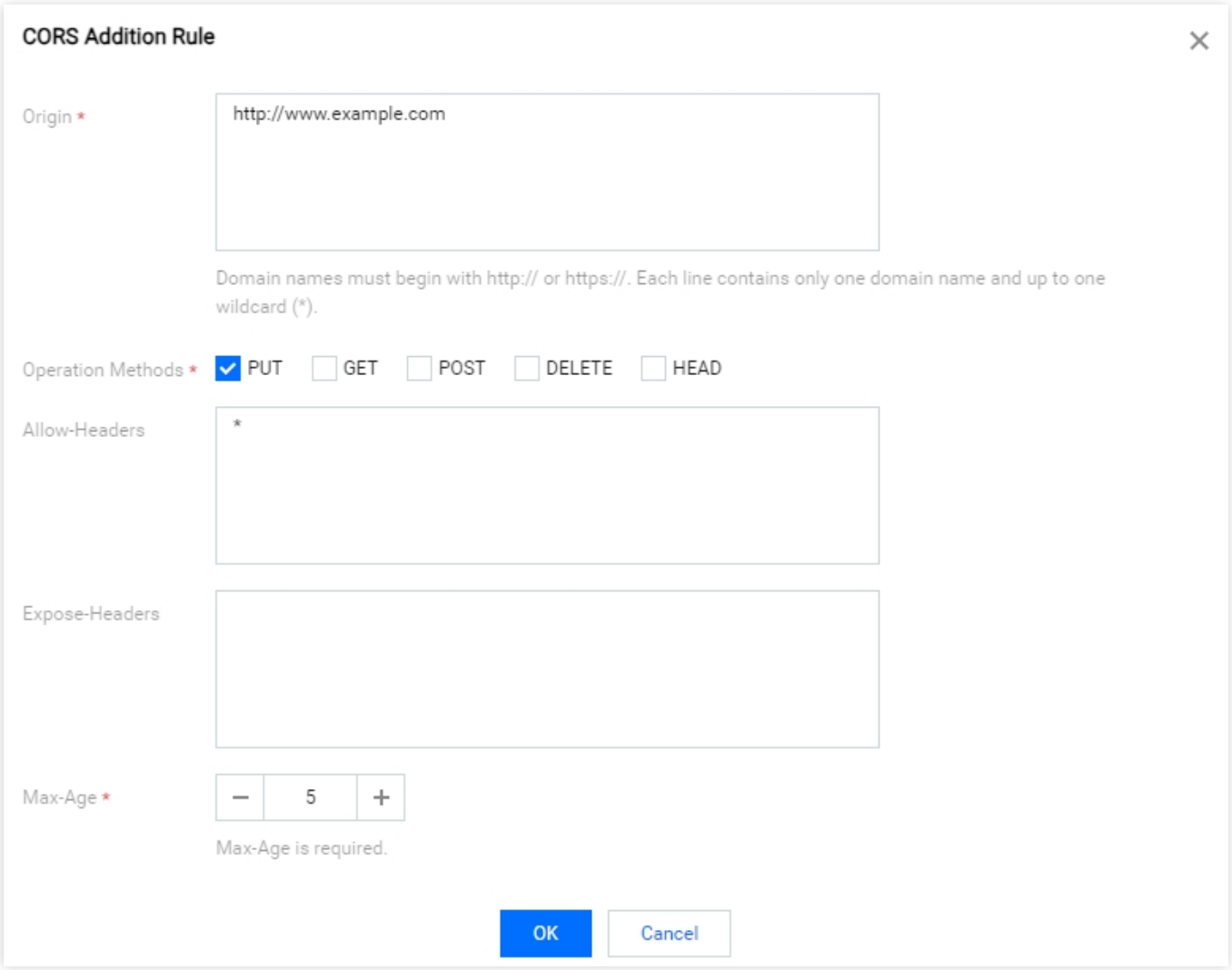
COS service supports response to OPTIONS requests for cross-origin access and returns specific rules set by developers to browsers. However, the server does not verify whether subsequent cross-origin requests conform to the rules. See the description of HTTP access control for more details.

Directions

1. Log in to the COS bucket console.
2. Select Bucket List in the left sidebar to go to the bucket list page.
3. Click the bucket for which you want to set cross-origin access to enter the bucket file list page.
4. Click Security Management > CORS Settings and click Add Rule.



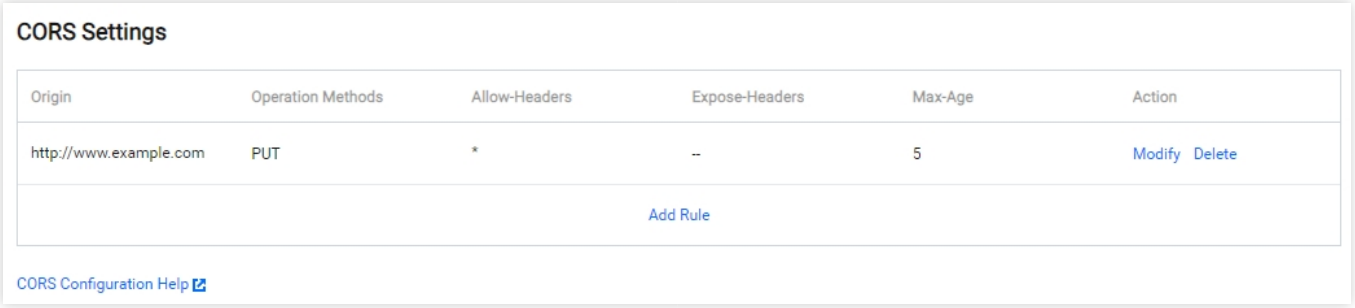
5. Add rule information (items marked with * are required). The configuration items are described as follows:



- Origin: The origins from which cross-origin requests are allowed.
  - You can specify multiple sources at the same time, but only one source can be entered per line.
  - You can set a * , indicating that all domains are allowed, which is not recommended.
  - A single specific domain name is supported, such as http://www.abc.com .
  - Second-level wildcard domain names, for example, http://*.abc.com , are supported. Note that each line can contain only one * .
  - Pay attention not to omit the protocol name http or https. If the port is not the default port 80, you also need to add a port.
- Operation Methods: GET, PUT, POST, DELETE, and HEAD are supported. Enumeration of one or more methods is allowed for a cross-origin request.
- Allow-Headers: When sending an OPTIONS request, tell the server which custom HTTP request headers can be used in the following request, for example, x-cos-meta-md5.
  - You can specify multiple Headers at the same time, but only one Header can be specified per line.
  - Header is easy to be omitted. If there is no special requirement, you are advised to set this field to * , meaning that all headers are allowed.
  - Uppercase and lowercase letters [a-z, A-Z] are supported, and no underscores (_) are allowed.

        - Each Header specified in Access-Control-Request-Headers must correspond to a value in Allowed-Header.
    - Expose-Headers: The Expose-Header returns the common Headers of COS. For details, see common request headers. Set this parameter according to the specific application requirements. Etag is recommended. Headers do not allow wildcards and are case-insensitive. You can set multiple headers, with one header per line.
    - Max-Age: Set the validity period (in seconds) of the result of the OPTIONS request. The value must be a positive integer, such as 600.
6. After the setting is completed, click OK.

   At this point, you can see that the cross-origin access rule has been added. If you need to modify, click Modify to edit your settings.



Setting a Static Website

Overview

You can use the COS console to set a bucket to host a static website and access the static website by accessing the bucket's static website domain name.

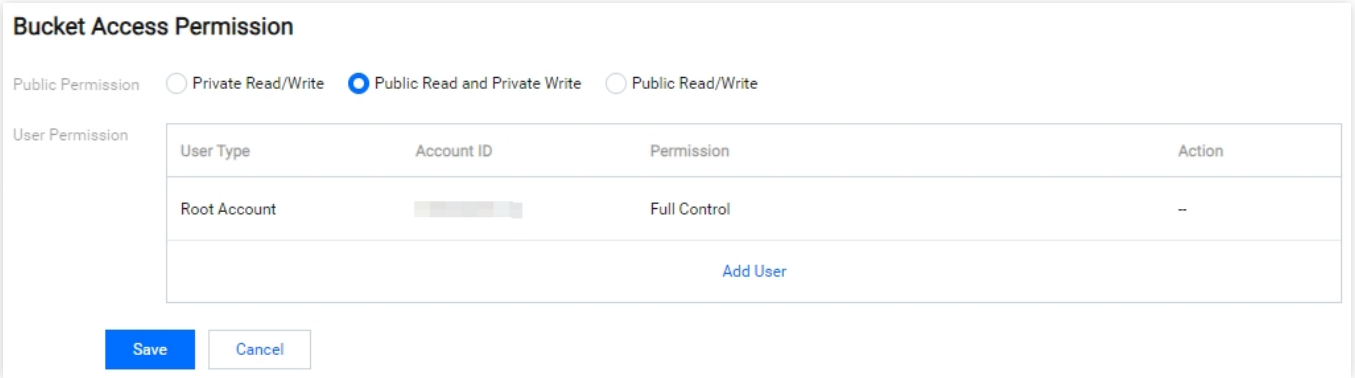
> Note:
>
> - To use a bucket to host a static website, you need to set the bucket's access permission to Public Read and Private Write.
> - After enabling the static website configuration, you need to use the static website domain name to access the COS origin server for it to take effect. If you use the COS default domain name to access, there is no static website effect.

Prerequisites

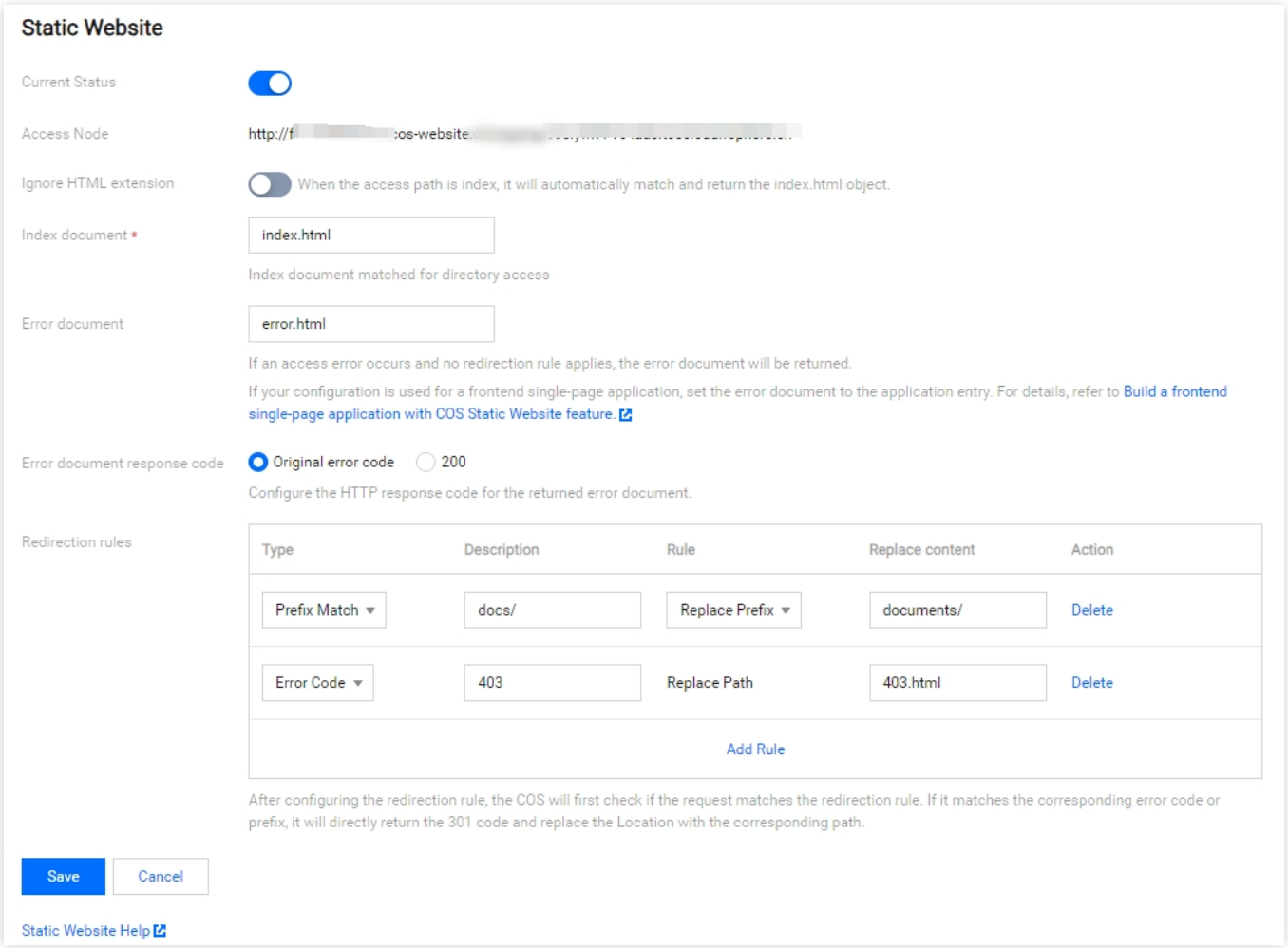
The bucket is created. For more information, see the document Create Bucket.

Directions

1. Log in to the COS console, click the Bucket List menu on the left, select the bucket you want to use to host the static website, and go to the bucket details interface.
2. Select Permission Management > Bucket Access Permission, select Public Read and Private Write in the Public Permission section, and save.



3. Select Basic Configuration > Static Website, click Modify, toggle on Current Status, and then fill in the configuration items of the static website one by one.



Description of the configuration is shown below:

- Ignore HTML extension (optional): If the access path is index, the index.html object will be automatically matched and returned.

- Index document (required): The index document is the homepage of the static website. It is the page returned when a user requests the root directory or any subdirectory of a website. This page is usually named index.html.

  > Note:
  >
  > If a folder is created in the bucket, the index document needs to be added at each level of the folder.

- Error document (optional): The error document is the page returned after an access error occurs at the static website. This configuration item allows you to customize your error document. When the static website fails to respond to the user's requests, the specified custom error page will be returned. For example, if you configure an error document named error.html, when the user encounters an HTTP error, the error.html page will be returned to help and guide the user. If you do not configure any error document, when the user encounters an HTTP error, a default error message will be returned.

> Note:
>
> Files in the bucket's root directory or subdirectories can be configured as error documents. Use a file in formats such as .html and .htm , which can be recognized by browsers. If you use a file in formats that cannot be recognized by browsers, for example, a .zip file, most browsers will display "Error, inaccessible" or "Access request denied."

- Redirect rules (optional): The redirect rules allow you to redirect requests based on specific file paths, prefixes in requests, or response codes.
  For example, if you delete or rename a file in the bucket, you can add a redirect rule to redirect access requests for this file to other files.
  - Error Code: Currently, the redirect rules only support redirect configuration for 4xx error codes (such as 404). You can customize the error page. If a corresponding HTTP error is triggered by a user, you can provide other guidelines for your user on this error page.
  - Prefix match: You can use the prefix matching rules to set redirects for files or folders in the bucket. For specific examples, see Redirect Rule Examples.

4. After completing the settings, click Save.

Sub-Account Access to Bucket List

Overview

Sub-account By default, there is no permission to pull the bucket list. Therefore, when you log in to the COS console using a subaccount, you cannot view the overview data, bucket list, and other management items that require authorization.
Sub-account Bucket lists can be accessed in two ways:

- By Add a Favorite Path: This method is for scenes where some sub-accounts have object operation permissions but do not have access to the bucket list. The access path can be a specific bucket or a directory under a bucket. Make sure the added path is authorized.
- By Add Preset Policy: You can add a preset policy QcloudCOSGetServiceAccess (obtain bucket list access permissions) for the sub-account through the root account to access the bucket list. In this method, you can also view the overview statistics on the console.

  > Note
  >
  > This feature applies to scenes where you log in to the console through a sub-account to access the bucket list.

Adding a Favorite Path

By default, the sub-account is not granted the preset policy (QcloudCOSGetServiceAccess). At this time, the sub-account has no permission to pull the bucket list. However, when the root account authorizes the sub-account user permissions (such as data reading, data writing, etc.) to a bucket, the sub-account can access the corresponding bucket through Add a Favorite Path.

Prerequisites

The sub-account has been authorized user permissions for a bucket by the root account. For details, see Set Access Permission.

Directions

1. Use sub-account to log in to the COS console and go to Favorite Path page, click Add a Favorite Path.
2. In the Add a Favorite Path pop-up, select the region where the bucket is located and input the favorites path. The configuration are described as follows:
   - Region: Select the region that corresponds to the bucket that you are authorized to access.
   - Favorites Path: Enter the name of the bucket that is authorized to access (for example,

examplebucket-1250000000), or enter the path of the object in the bucket, for example, examplebucket-1250000000/doc/exampleobject.txt .

3. After confirming that the region and favorites path are correct, click OK. Then, you can add the authorized bucket or the path of the object in the bucket.
4. Click File List on the right; you can see the objects to which the root account authorizes permissions to the sub-account.

Adding Preset Policy

Add Preset Policy QcloudCOSGetServiceAccess (obtain bucket list access permissions) for sub-account to access the bucket list.

> Note
>
> - The preset policies QcloudCOSFullAccess or QcloudCOSReadOnlyAccess can also authorize sub-accounts permission to access the bucket list. However, due to the wide range of permissions authorized by these two policies, for security reasons, their use is not recommended.
> - The overview statistics depend on the access permissions of the bucket list. When the sub-account needs to pull statistical data, make sure that the root account has added the preset policy QcloudCOSGetServiceAccess for the sub-account. Otherwise, it will prompt that you do not have permission to access the statistical data.

1. Log in to the CAM console with your root account and enter the user list page.
2. Find the sub-account to which you want to add the policy and click Authorization on the right to go to the Association Policy page.
3. In the policy list, search for and add the preset policy QcloudCOSGetServiceAccess (bucket list access permission for COS).
4. Click OK to complete the policy association.
5. Click the sub-account user name to enter the sub-account details page. You can see the added policies here. You can unbind the policy when you no longer need to use it.

   > Description
   >
   > At this point, you have added a preset policy to the sub-account through the root account. Now, you can log in to the COS console through the sub-account and view the bucket list and overview statistics normally.

Adding Bucket Policies

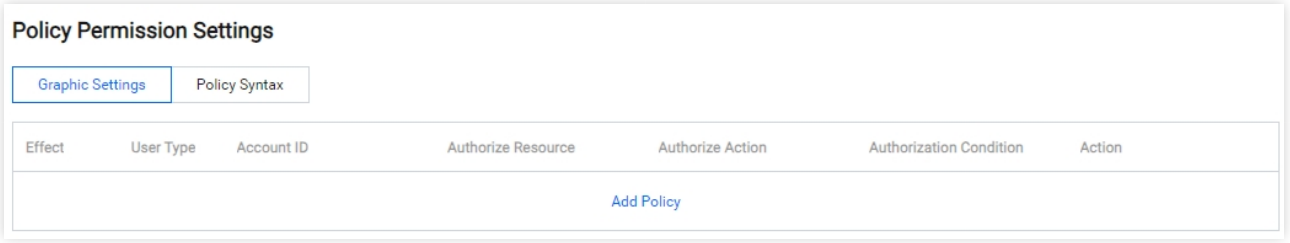
Overview

You can use the COS console to add policies for a bucket to allow/deny an account or a source IP address (or IP range) access to the COS resources set by the policies. The following describes in detail how to add bucket policies.

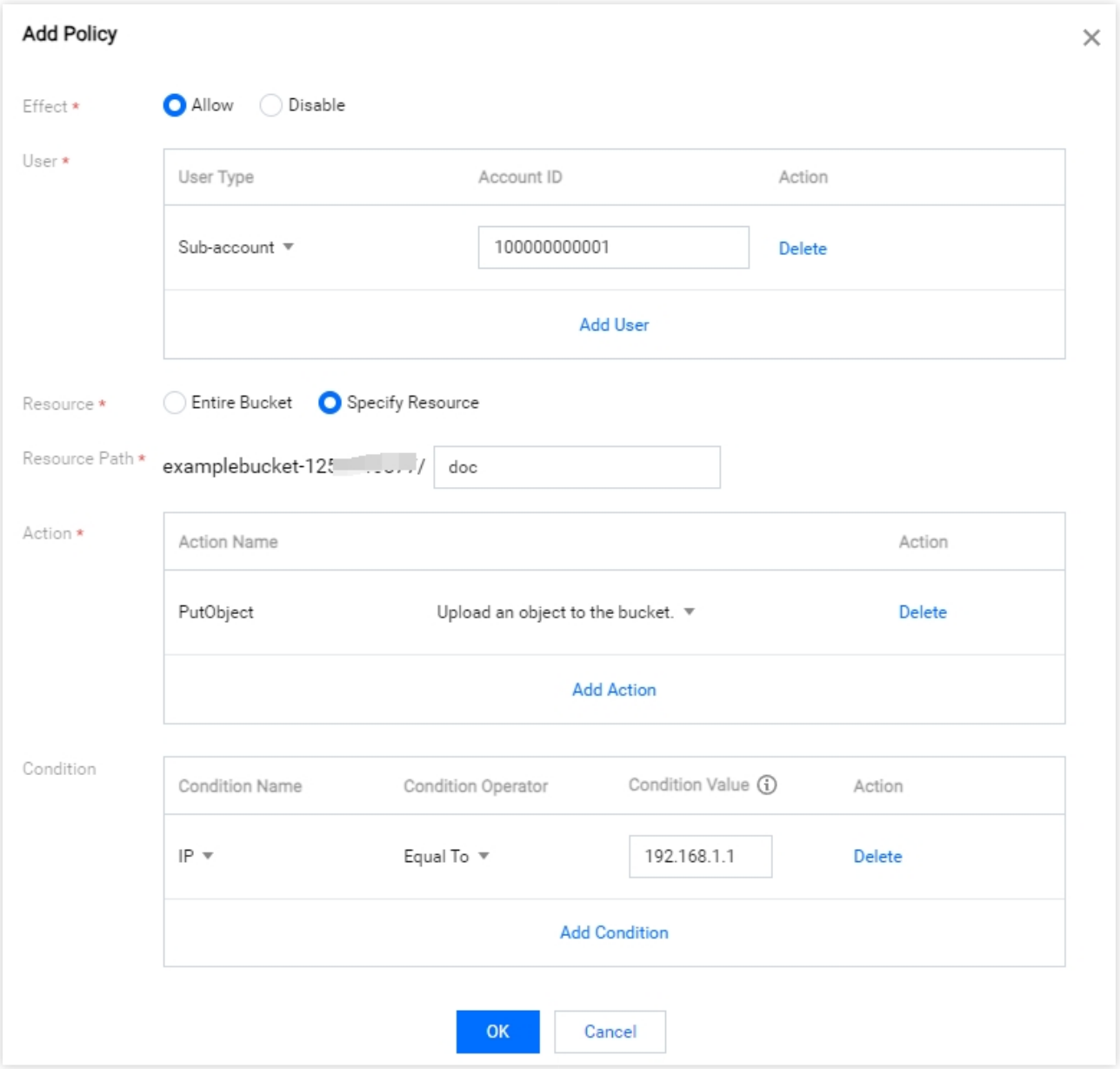
> Note:
>
> For each root account, the total number of object ACL, bucket ACL, and bucket policies that can be created is up to 1,000.

Directions

1. Log in to the COS console.
2. In the left sidebar, click Bucket List.
3. Select the bucket you want to add a bucket policy to go to the bucket page.
4. Select Permission Management > Policy Permission Settings. COS provides two ways to add bucket policies: Graphical Settings and Policy Syntax. You can choose one of them to add bucket policies.

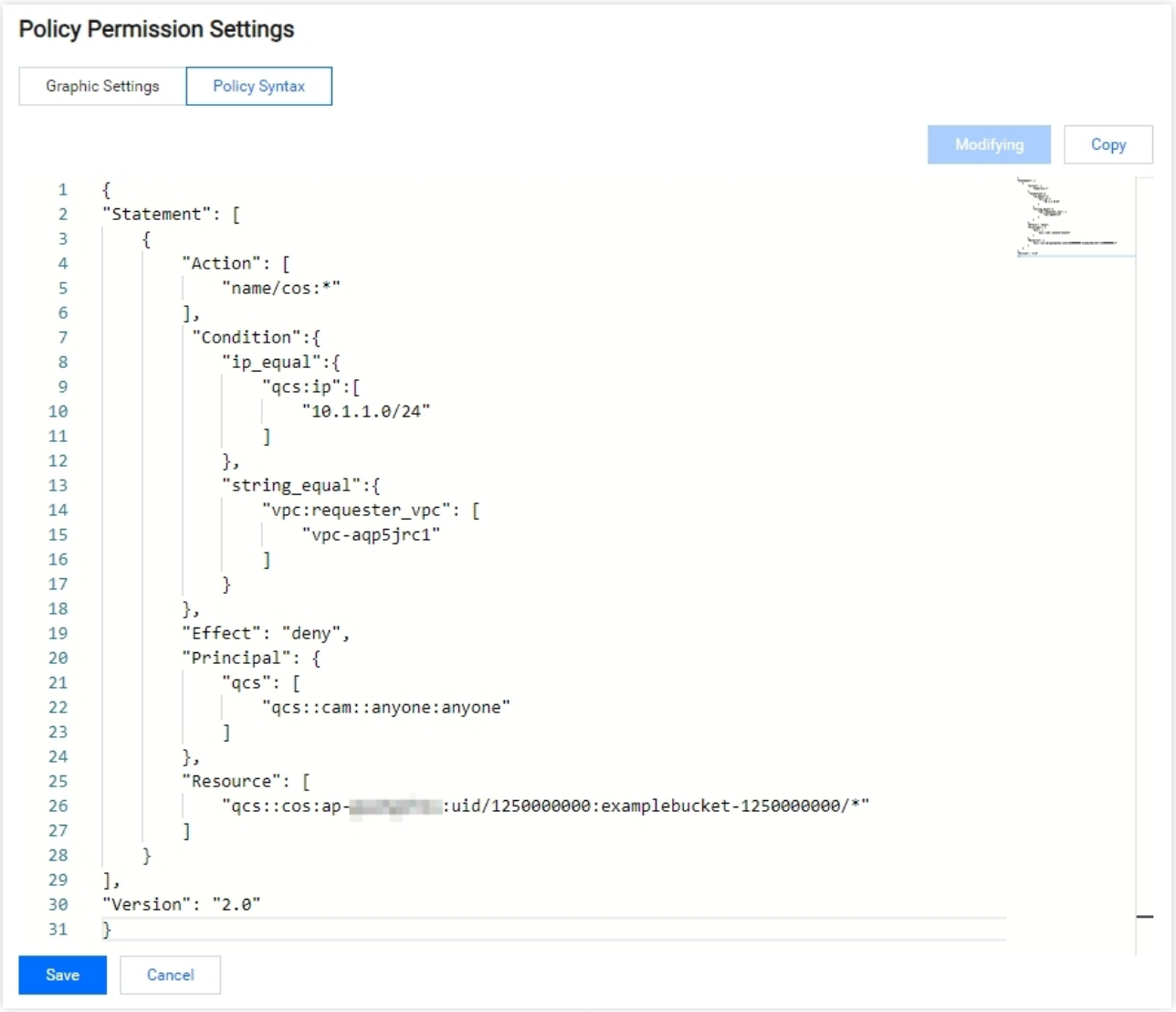


   - Graphical Settings
     The sample settings are shown below:



- Policy Syntax

  Click Modify and type the policy syntax you define.



5. After confirming that the configuration is correct, click Save. Then, if you use a sub-account to log in to the COS Console, you can only access the resources set by the policy.

Setting Inventory

Overview

You can use the COS console to enable the inventory feature for your buckets. Enabling the inventory feature allows you to regularly (daily/weekly) export an inventory report about information such as the object attributes and detailed configurations. For more information about inventory, see Inventory Overview. The following describes in detail how to enable the inventory feature for a bucket. Both scheduled and interim inventories are supported. You can choose one as needed.

> Note
>
> - Multiple inventory tasks can be configured for a single bucket.
> - The object content is not directly read during the execution of inventory tasks, and only the attribute information such as the object metadata is scanned.

Prerequisites

The bucket is created.

Directions

Adding a Scheduled List

1. Log in to the COS console.
2. In the left sidebar, click Bucket List.
3. Click to enter the bucket (source bucket) for which you want to enable the inventory feature.
4. In the left sidebar, select Basic Configuration > Inventory Settings, and click Add Inventory.
5. On the configuration page, you can configure the following:
    i. Basic Information
        - Rule Status: Select whether to enable the current inventory rule. You can select to enable/disable the inventory.
        - Inventory Name: Type the name of your inventory.
        - Target bucket: Refers to the bucket where the inventory is stored. The initial default destination bucket is the source bucket. The destination bucket must be in the same region as the source bucket.
        - Inventory report prefix (optional): Input a prefix of your choice for the target storage bucket, which can be used to group inventory files in a common location. The initial value

is the default value.

ii. Filter Conditions

- File range: Select the scope of the inventory objects. You can select the entire bucket or some files with the same prefix. For example, to select all files in the doc path under the folder path in the bucket, specify the file prefix as folder/doc .
- Object Version: Select whether the inventory will include all versions of the object or only the current version. If this is not selected, only the current version will be kept by default.
- Object Tag: List objects with the same tag to the inventory through filtering. If this is left blank, no filtering will be performed by default.
- Filtering Time (optional): List objects whose modification time is after a certain time point or within a certain time to the inventory through filtering. If this is not set, no filtering will be performed by default.
- Custom output header (optional): If you need to include custom headers of the object in the inventory report, you can fill in the custom headers to be output. Only x-cos-meta-* headers are supported. If not filled in, the custom header will not be output by default.
- Inventory Information: Select the corresponding object information to be included in the inventory report. The options are object size, storage type, ETag, bucket replication status, multipart upload status, last update date, tag, and CRC64. If this is not selected, all options will be selected by default.

> Description

- ETag (Entity Tag) is a hash of an object. ETag reflects only changes to the content of an object but not changes to its metadata. The ETag is not necessarily an MD5 digest of the object's data, depending on how the object was created and encrypted.
- By default, the generated inventory content contains the Appid, Bucket, Key, and LastModifiedTime fields.
- If versioning is enabled for the bucket, the generated inventory contents have the Version ID, IsLatest, and IsDeleteMarker fields added.

3. Output Format
   - Output format: The output inventory file format is a CSV file compressed using GZIP.
   - Generation Cycle: Select whether to generate the inventory daily/weekly/monthly. The default is to generate the inventory daily. The specific time to generate the inventory depends on the amount of data. The description of the configuration is shown below:
     - Daily: For example, if an inventory is added at 15:00 p.m., the inventory file will usually be generated and delivered to the target storage bucket at 6:00 a.m. the next day at the latest.
     - Weekly: By default, the inventory is generated in a 7-day cycle. For example, if you start the inventory on September 1st, the first inventory will be generated on September 2nd,

and then on September 9th, September 16th, September 23rd, and so on, with a 7-day cycle.

- Inventory Encryption: Select whether the inventory needs server-side encryption. The options are:
  - No encryption: The inventory will not be encrypted. This is the default option.
  - SSE-COS: Encrypt reports using server encryption with COS-managed keys.
- Access authorization: Disabled by default. After enabling access authorization, you can proceed to the next step.

4. Confirmation of Information
   Confirm the inventory configuration information for the bucket. If you need to modify, click Previous.
5. After confirming that the configurations are correct, click OK. Afterward, COS will generate inventory files daily or weekly and ship them to the destination bucket you set.

> Description
>
> For details on the format and content of the inventory report generated by the inventory feature, see Feature Overview of Inventory.

Generating Instant Inventory

1. Log in to the COS console.
2. In the left sidebar, click Bucket List, and then click to enter the bucket (source bucket) for which you want to enable the inventory feature.
3. Click Basic Configuration > Inventory Settings on the left and select an inventory rule. Click Generating inventory immediately on the right to generate an inventory instantly.

> Description
>
> - If you have not configured an inventory, configure an inventory first to generate an instant inventory.

Setting Bucket Tags

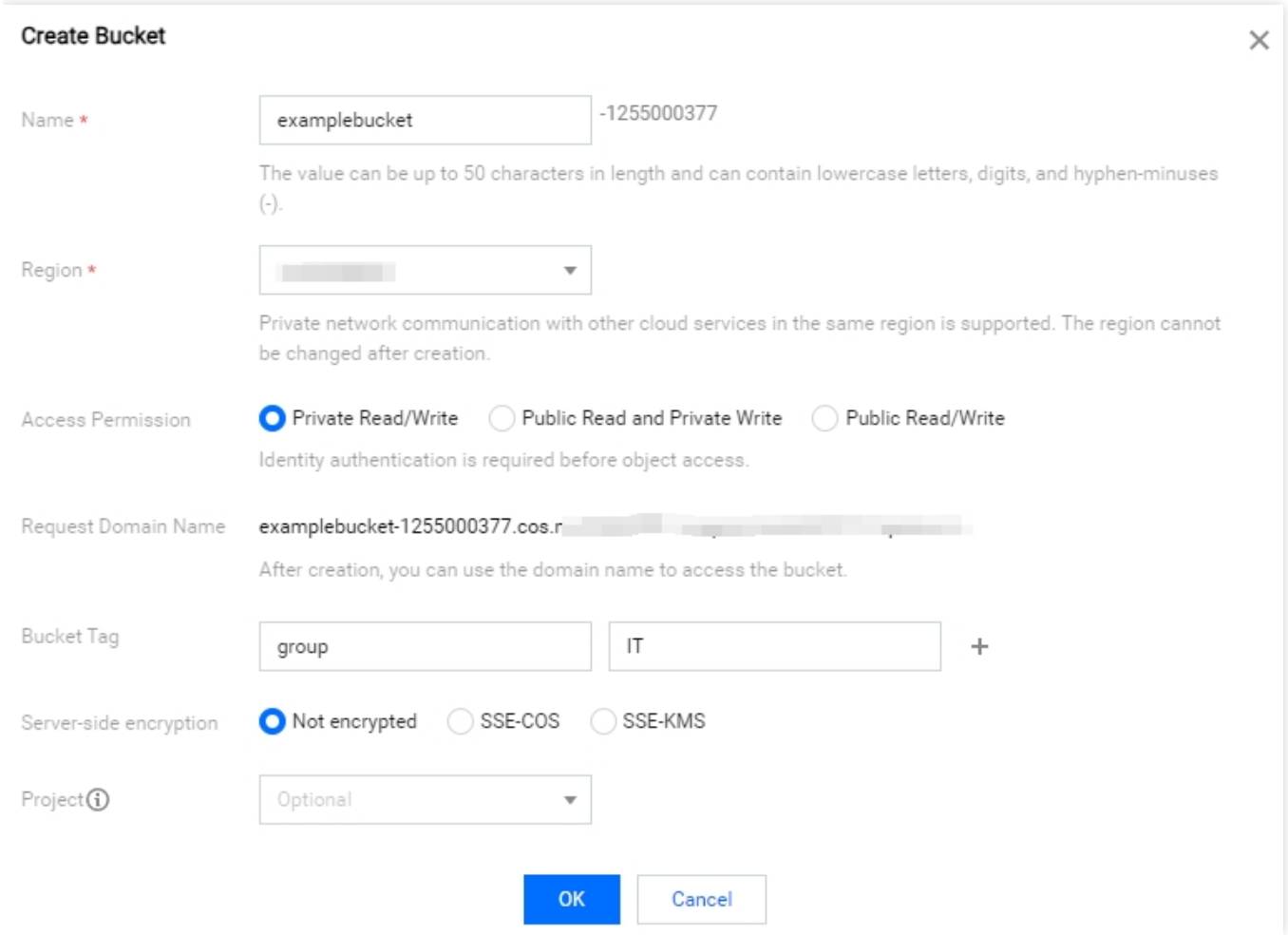
Overview

A bucket tag is a key-value pair (key = value), which is comprised of a tag key and a tag value that is connected by an equal sign =, for example, group = IT. It can serve as a marker for managing buckets, facilitating users in grouping and managing buckets. t can serve as a marker for managing buckets, facilitating users in grouping and managing buckets.

> Note:
>
> - A bucket can have a maximum of 50 tags, and tag keys cannot be repeated.
> - Tag keys and tag values must not use qcs: or project reserved fields.

Adding Tags When Creating a New Bucket

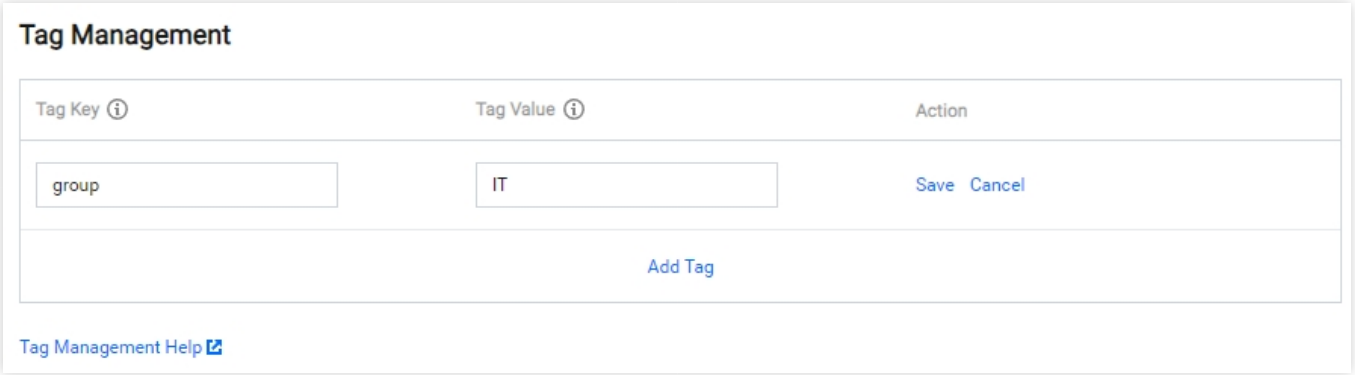
You can add a bucket tag when you click Create Bucket, as shown in the following figure:



Adding Tags to the Created Bucket

If you did not add tags when you created the bucket, you can follow the steps below to add tags to the bucket.

1. On the bucket list page, find the bucket to which you want to add a tag and click its name to go to the bucket configuration page.
2. Click Basic Configuration > Tag Management and click Add Tag to add a bucket tag.



3. After entering the tag key and tag value, click Save to complete adding a bucket tag.

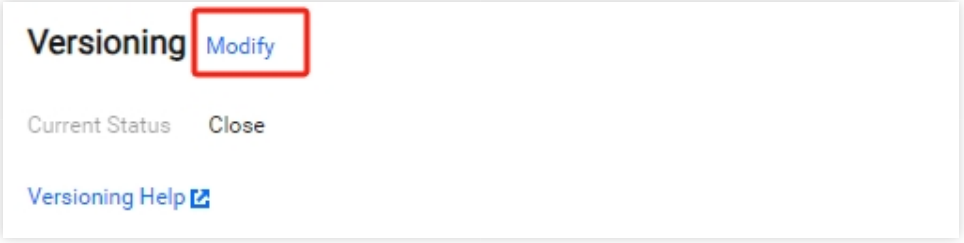
Setting Versioning

Overview

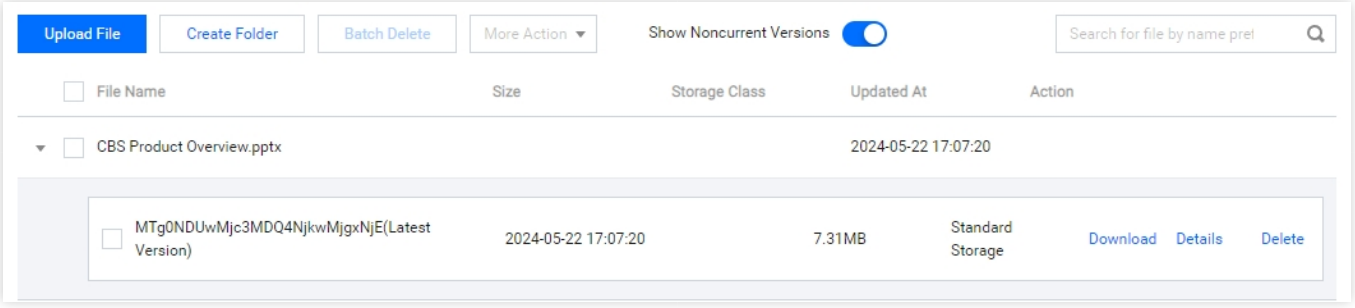
Enable bucket versioning to store multiple versions of objects in a bucket and retrieve, delete, or restore objects of a specified version. This helps recover data that was accidentally deleted by the user or lost due to application failure.

Directions

1. Log in to the COS console.
2. In the left sidebar, select Bucket List to enter.
3. Locate the bucket for which you want to configure a version, and click its name to enter the bucket list.
4. Select Fault Tolerance and Disaster Recovery Management > Versioning and click Modify to enter the editing status.



5. Click Enabled button of Current Status and save.
6. In the pop-up dialog box, click OK to enable versioning.
   When you do not need to use the versioning function, change the current On button to Off.
7. After enabling versioning for a bucket, if you upload a file whose name already exists in the bucket, you can click Show Noncurrent Versions on the file list interface to view this file's previous versions uploaded at different time points. You can also perform various operations on the latest version and historical versions, and delete marker versions.
   Historical version files support Download, Details, Delete operations.



Setting Bucket Replication
Replication Within the Cloud

Overview

After enabling cross-region replication, you can copy incremental objects automatically and asynchronously from the source bucket to a target bucket in another region. When you perform management operations on objects in the source bucket (such as adding objects or deleting objects), COS will automatically copy these operations to the target bucket. You can choose to enable or disable cross-region replication based on your actual needs.

Prerequisites

Before enabling inter-cloud replication, ensure that the source bucket and the target bucket are in different regions, and have the Enable Version Control feature activated.

Enabling Inter-Cloud Replication

1. Log in to the COS console.
2. In the left sidebar, select Bucket List to enter.
3. Click the name of the bucket for which you want to configure inter-cloud replication to go to the bucket's file list page.
4. Select Fault Tolerance and Disaster Recovery Management > Bucket Replication, in the Inter-Cloud Replication configuration, click Add Rule to enter the inter-cloud replication rule configuration status.
5. Configuring inter-cloud replication rules requires that both the source bucket and the target bucket have versioning rules enabled. If the version control feature is not enabled on both the source bucket and the target bucket, enable the versioning feature before configuring the inter-cloud replication rule.
6. After the rules setting is completed, click OK.
   The configuration items are described as follows:
   - Source Region: The region to which your source bucket belongs.
   - Application Scope: Refers to the range of objects in the source bucket that need to be copied. If not set, all objects in the bucket are copied by default. If a prefix is specified, the content with a specific prefix is copied (for example, to copy a file with the logs/ prefix, fill in logs/).

- Resource Path: Your source bucket path.
- Destination Bucket: Refer to the bucket where the copied object is stored. Its region cannot be the same as the region of the source bucket. It does not support selecting a bucket that is not in the current account.
- Destination Storage Class: The storage type of objects after they are replicated to the destination bucket. This item follows the object type in the source bucket by default. You can select another storage type to modify the storage type of the object copies in the destination bucket. Currently, two storage types are supported: STANDARD and STANDARD_IA.

> Note:
>
> - After you have configured the rules, you can manage them. You can click the button to enable or disable the current rule and click the Edit button to modify the current rule.
> - If you set the scope of inter-cloud replication rules to replicate all content in the storage bucket during the initial setup, you will not be able to add any new rules. You can modify the current rule by editing it or deleting it and then adding it again.
> - If you set the application scope to the contents of a certain prefix when you first set up the inter-cloud replication rule, you can still modify the application scope to the entire content in the bucket by editing the current rule.

Disabling Inter-Cloud Replication

You can disable inter-cloud replication by disabling the status button or by deleting the rule.

- Disable status button: Temporarily disable the current rule by turning off the status button in a rule. This operation will pause the current inter-cloud replication feature. The replicated data will be retained in the target bucket, and the incremental data in the source bucket will no longer be copied to the destination bucket. If you need to use it again, click the Enable button.
- Delete rule: Delete the added rules under the Inter-Cloud Replication management item. After deleting the rules, the configured inter-cloud replication rules will become invalid. The replicated data will be retained in the Target bucket, and the incremental data in the source bucket will no longer be replicated inter-cloud. If you need to use it again, you need to add the rules again.

> Note:
>
> - Inter-cloud replication operations that have not yet been completed will be aborted when inter-cloud replication is disabled and will not be able to continue.
> - When you enable inter-cloud replication for a bucket again, inter-cloud replication is performed only on objects that were added after the enabling.

Setting Cross-Cloud Replication

Overview

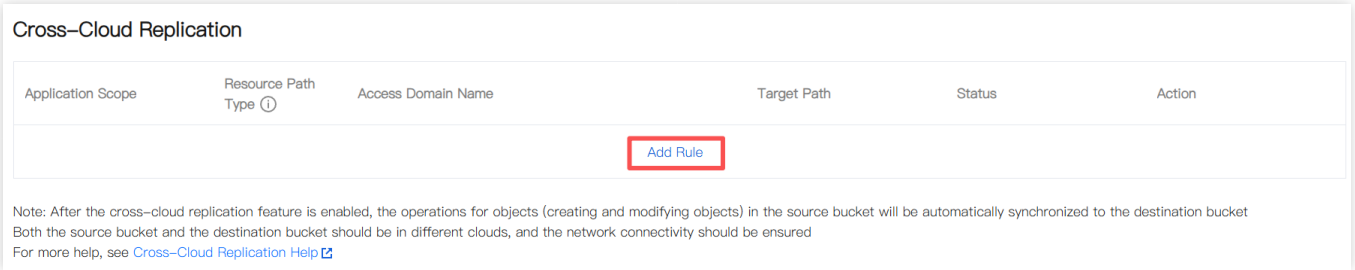
After enabling cross-cloud replication, you can automatically and asynchronously copy incremental objects from the source bucket to another target bucket that supports the S3 protocol. When you perform management operations on objects in the source bucket (such as adding objects or deleting objects), COS will automatically copy these operations to the target bucket. You can choose to enable or disable cross-cloud replication based on your actual needs.

Prerequisites

Before enabling the cross-cloud replication feature, ensure that the source bucket and the target bucket have connected network status and you can obtain the permission information of the target bucket.

Enabling Cross-Cloud Replication

1. Log in to the COS console.
2. In the left sidebar, select Bucket List to enter.
3. Click the name of the bucket for which you want to configure cross-cloud replication to go to the bucket's file list page.
4. Select Fault Tolerance and Disaster Recovery Management > Bucket Replication. In the Cross-Cloud Replication configuration, click Add Rule to enter the cross-cloud replication rule configuration status.



5. Fill in the corresponding configuration information according to the configuration interface.
   The parameters are described as follows:
   - Source Region: The region to which your source bucket belongs.
   - Application Scope: Specify the source bucket resources for the cross-cloud replication rule.
   - Resource Path: After selecting designated resources, you can configure specific resource prefix information. If a prefix is specified, copy content with a specific prefix (for example, to copy files with the logs/ prefix, enter logs/).
   - Access Domain Name: Fill in the domain information of the target bucket, and include the http://

information.
- Destination Bucket AK: Access permission information for the target bucket.
- Destination Bucket SK: Access permission information for the destination bucket.
- Destination Bucket: Information of the target bucket, formatted as bucket_name-appid, supporting any COS type compatible with the S3 protocol.

> Note:
>
> - After you have configured the rules, you can manage them. You can click the button to enable or disable the current rule and click the Edit button to modify the current rule.
> - If you set the scope of cross-cloud replication rules to replicate all content in the storage bucket during the initial setup, you will not be able to add any new rules. You can modify the current rule by editing it or deleting it and then adding it again.
> - If you set the application scope to the contents of a certain prefix when you first set up the cross-cloud replication rule, you can still modify the application scope to the entire content in the bucket by editing the current rule.

Disabling Cross-Cloud Replication

You can disable cross-cloud replication by disabling the status button or by deleting the rule.

- Disable status button: Temporarily disable the current rule by turning off the status button in a rule. This operation will pause the current cross-cloud replication feature. The replicated data will be retained in the target bucket, and the incremental data in the source bucket will no longer be copied to the destination bucket. If you need to use it again, click the Enable button.
- Delete rule: Delete the added rules under the Cross-Cloud Replication management item. After deleting the rules, the configured inter-cloud replication rules will become invalid. The replicated data will be retained in the Target bucket, and the incremental data in the source bucket will no longer be replicated inter-cloud. If you need to use it again, you need to add the rules again.

> Note:
>
> - Cross-cloud replication operations that have not yet been completed will be aborted when cross-cloud replication is disabled and will not be able to continue.
> - When you enable cross-cloud replication for a bucket again, cross-cloud replication is performed only on objects that were added after the enabling.
> - Newly configured and re-enabled cross-cloud replication rules may have an effective time of up to 1 minute. Uploaded objects during this 1 minute may not be synchronized to the destination bucket.
> - Cross-cloud replication performs inventory recompilation operations at midnight every day and copies objects that were not synchronized or failed to synchronize on the day again.

Setting Bucket Encryption

Overview

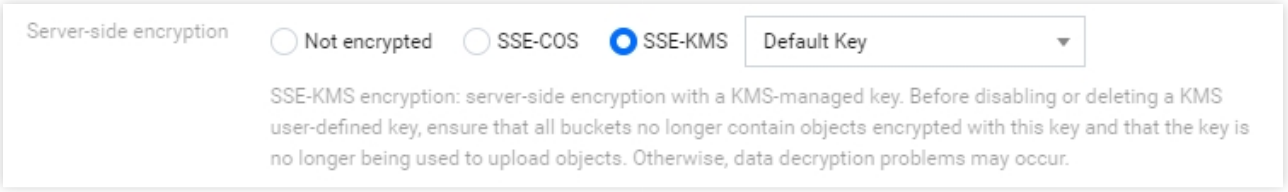
With the COS console, you can set server bucket encryption and encrypt all new objects uploaded to the bucket by default.
Currently, the bucket encryption method supports SSE-COS encryption (server-side encryption with COS-managed keys) and SSE-KMS encryption (server-side encryption with KMS-managed keys).

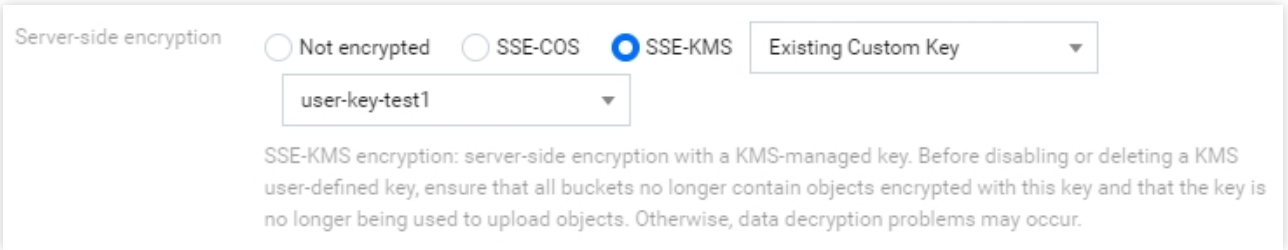
Directions

Setting Encryption When Creating a Bucket

When you create a bucket, you can set encryption for the bucket by following the steps below.

1. In the bucket list, click Create Bucket to fill in the basic settings, such as name and region.
2. In the Server-side encryption option, select SSE-COS or SSE-KMS.
   - When SSE-COS is selected, the encryption algorithm will display AES256. If the SM4 algorithm encryption function is enabled, the encryption algorithm will also display SM4.
   - When SSE-KMS is selected, the encryption algorithm is displayed as above, and the key can be selected as Default Key and Existing Custom Key.





3. Click OK.

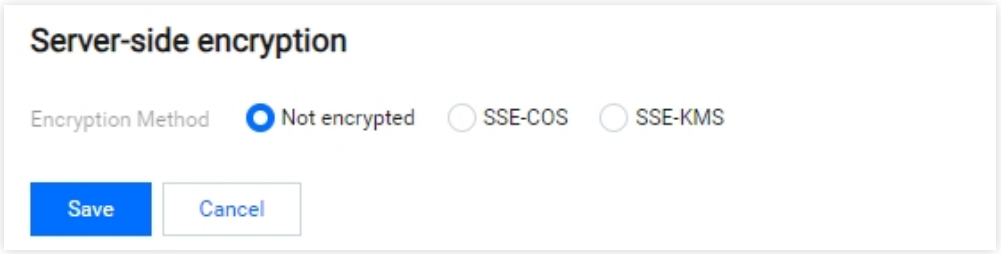
Setting Encryption in a Created Bucket

If you did not set encryption when you created the bucket, you can follow the steps below to set encryption for the bucket.

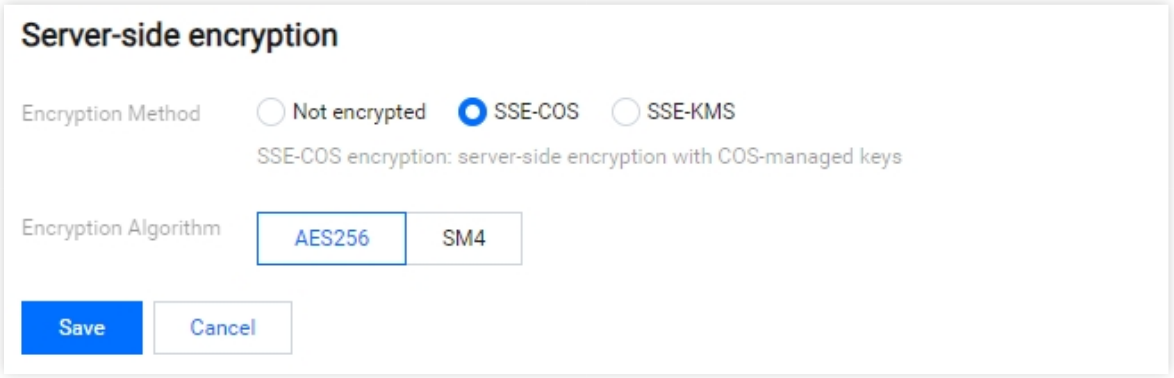
1. On the bucket list page, find the bucket for which you want to set encryption and click its name to go to the bucket configuration page.
2. Select Security Management > Server-side Encryption and click Modify to modify the encryption attributes of

the current bucket:

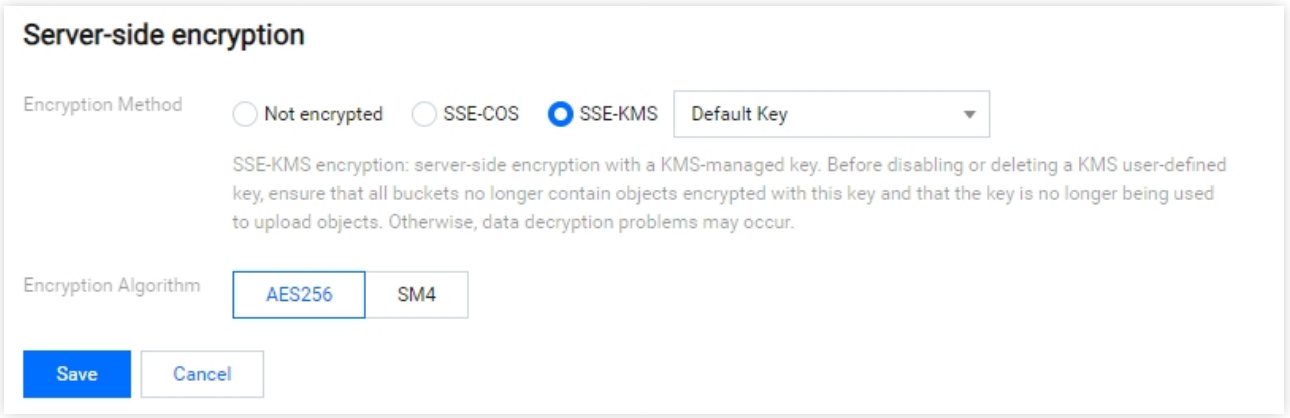
Unencrypted:



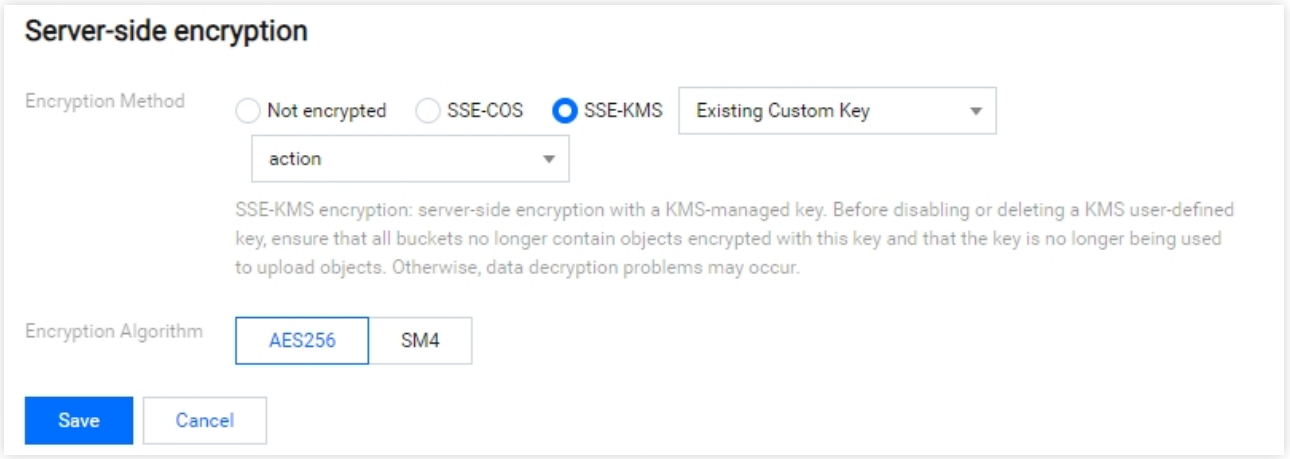
SSE-COS encryption:



SSE-KMS encryption (using default keys):



SSE-KMS encryption (using user-defined keys):



When you choose to use a user-defined custom key, avoid disabling or deleting it. Otherwise, unexpected encryption errors may be caused or the encrypted objects may be unable to be decrypted.

Problems of Encrypting Storage Buckets and Uploading Objects

There are two ways to encrypt objects when uploading. One is to add the corresponding header x-cos-server-side-encryption when uploading objects (choosing the encryption method when uploading objects on the console belongs to this category). The other is to set bucket encryption.
The priority of these two encryption methods is as follows:

1. When encryption is not set for the bucket, encryption with the header carried when the object is uploaded takes precedence. Then, the encryption can be "No Encryption", "SSE-COS", or "SSE-KMS".
2. When encryption is set for the bucket, and the corresponding header is not carried when the object is uploaded ("No Encryption"), then bucket encryption takes precedence and can be "SSE-COS" or "SSE-KMS".
3. When encryption is set for the bucket, and the corresponding header is carried when the object is uploaded ("SSE-COS" or "SSE-KMS"), then encryption with the carried header takes precedence, no matter what encryption type is set for the bucket encryption.
   Select the specified encryption method, and then click Save. The bucket encryption configuration is complete.

Configuring Custom Domain

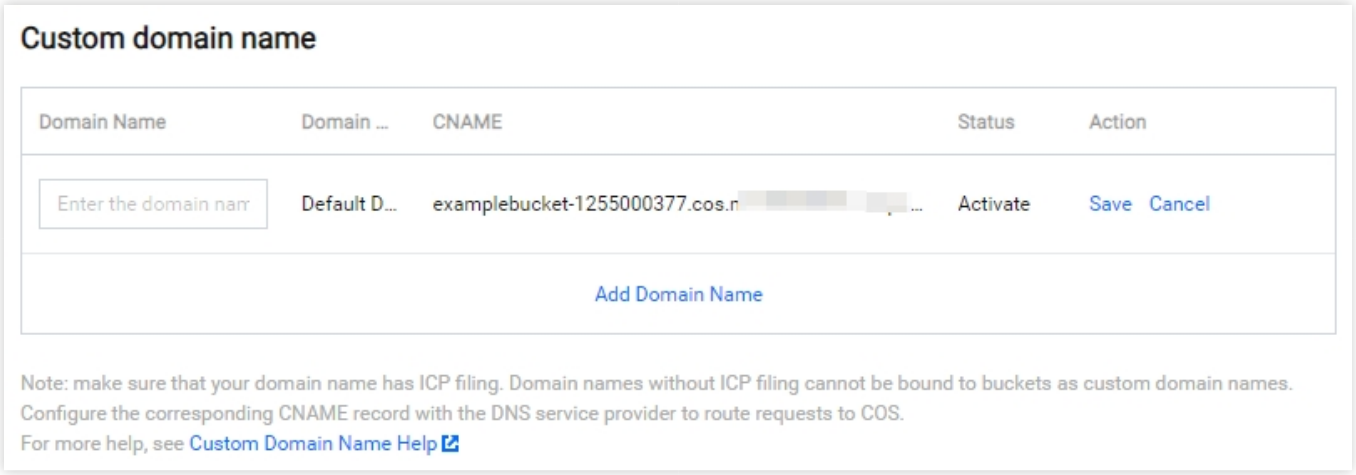
Overview

Users can bind a registered custom domain name to the current bucket and access objects in the bucket through the custom domain name.

> Note:
>
> The maximum number of custom domain names that can be added through the COS console is 20.

Directions

1. Log in to the COS console.
2. In the left sidebar, click Bucket List to go to the bucket list page.
3. Click the bucket whose domain name you want to configure to go to the bucket configuration page.
4. Select Domain Name Management and click Add Domain Name. If your custom domain name has been registered with the Ministry of Industry and Information Technology and can be CNAMEd to the default domain name of the bucket. Directly input the custom domain name in the Domain Name input box and click Save.



If you have not yet added resolution for your custom domain name, modify the CNAME information at your DNS vendor.

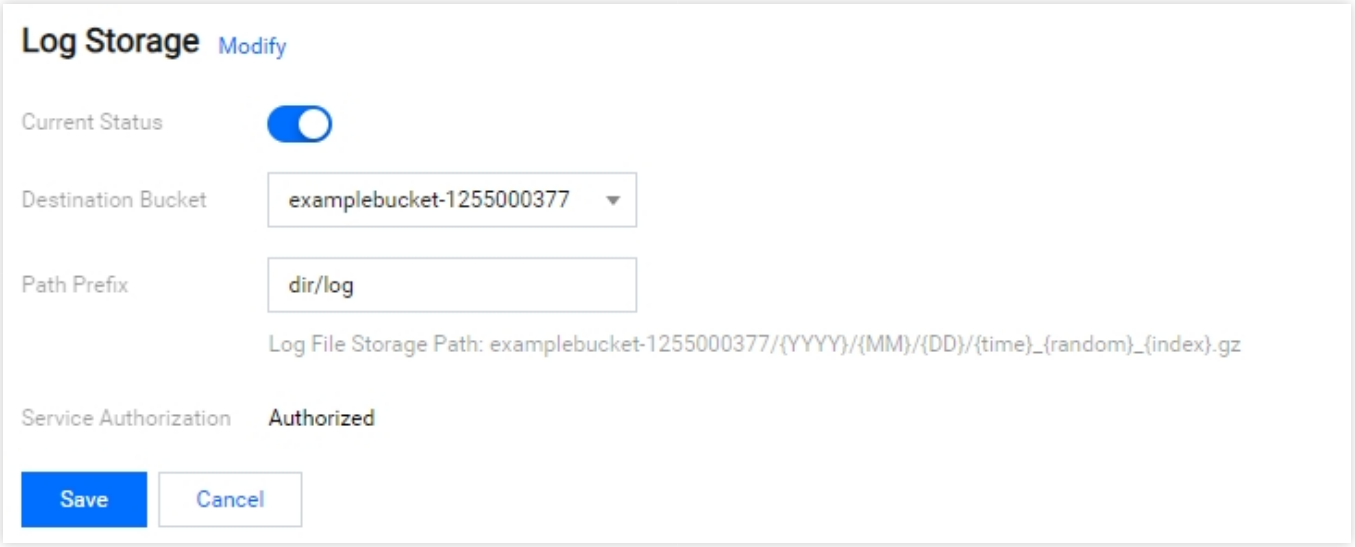
Setting Log Management

Overview

The log management feature allows you to record various request logs related to bucket operations. Enabling the log management feature can help you better manage and use your buckets.

Directions

1. Log in to the COS console.
2. In the left sidebar, click Bucket List to go to the bucket list page.
3. Find the source bucket that you want to enable log management for and click the bucket name to go to the bucket management page.
4. Select Log Management, click Modify, and toggle on the "Enabled" status.



   - Destination Bucket: The source bucket with enabled log management and the destination bucket for storing logs should be in the same region. We do not recommend that you set the source bucket as the bucket for storing logs.
   - Path Prefix: Type a custom path prefix to make it easier for you to find logs. If it is left blank, it will be the root path of the destination bucket by default.
   - Service Authorization: Authorize the CLS product services to ship access logs to your bucket.
5. After confirming that the information is correct, click Save. Then, the log management feature is enabled.

> Note:
>
> It might take several minutes or longer for the logs to be generated and shipped to the destination bucket. Please wait for a moment.

6. Find the destination bucket you configure for storing logs, and you will see the generated log files.

7. After downloading the log files, you can view them according to the field description.

Setting Event Notifications

Overview

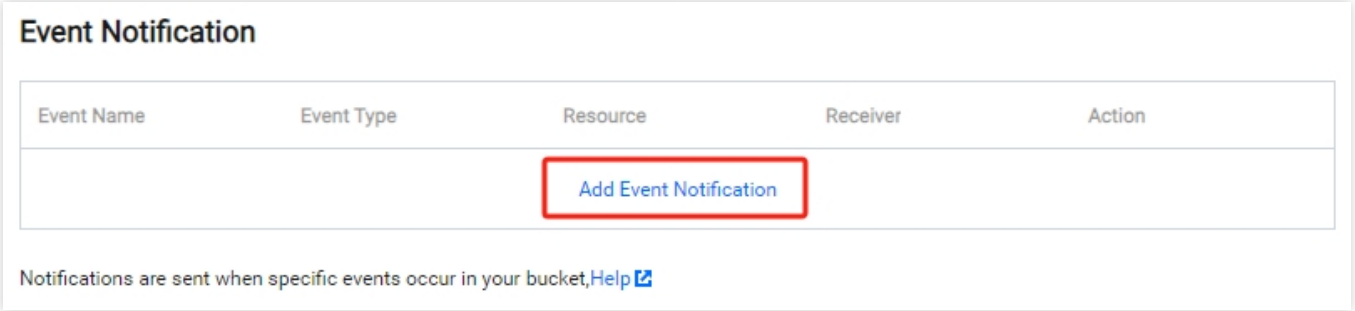
The event notification feature enables prompt notification for user-concerned resource operations within the bucket.

Prerequisites

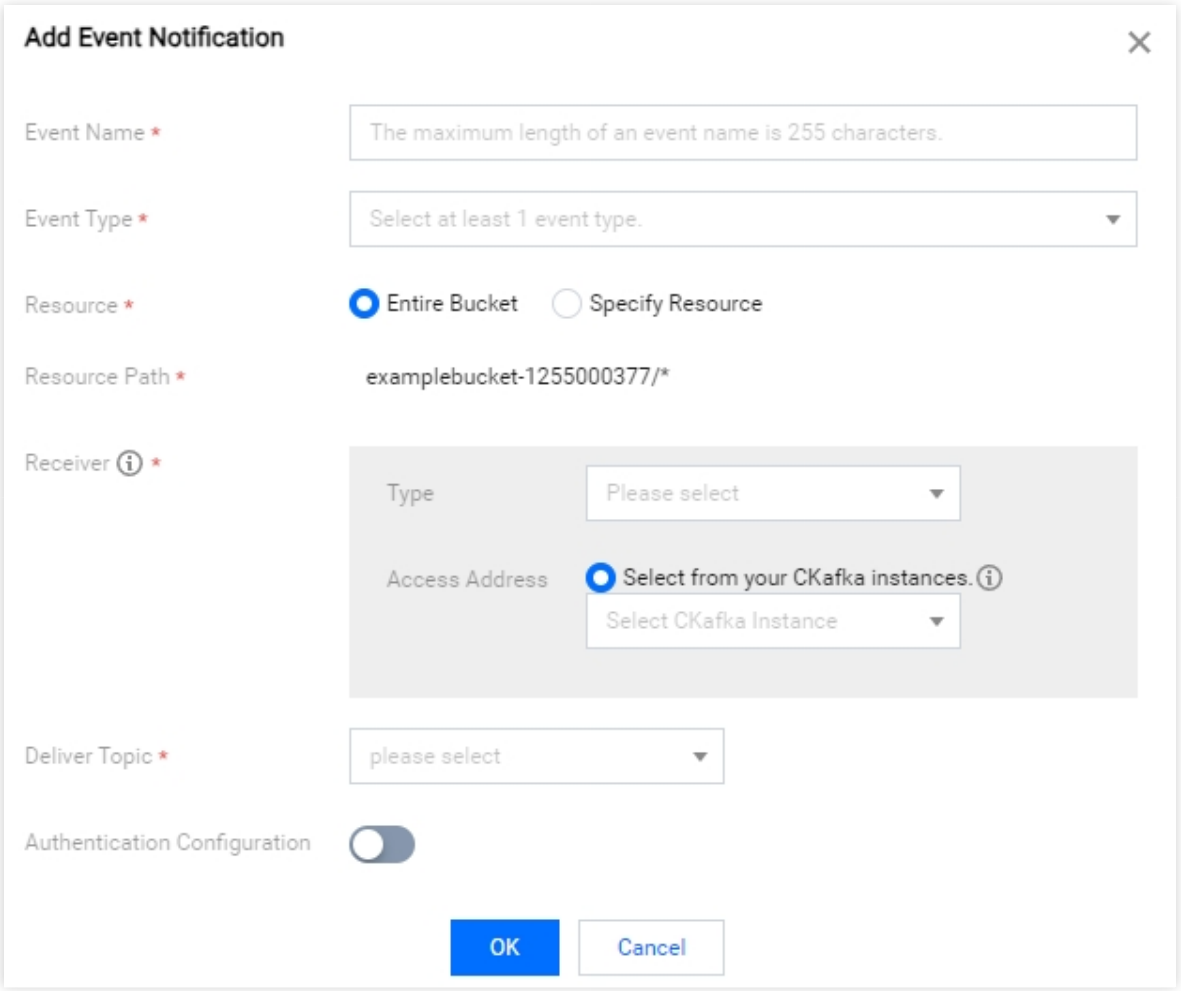
The Ckafka instance has been created, and a topic has been configured.

Directions

1. Log in to the COS console.
2. In the left sidebar, click Bucket List to go to the bucket list page.
3. Find the source bucket for which you want to set event notifications, click the bucket name, and go to the bucket management page.
4. Select Basic Configuration > Event Notification and click Add Event Notification.



5. You can customize event notification rules.



6. After the settings are completed, click OK to see the event notification rules. If you need to modify, click Modify to edit your settings.

Object Management
Uploading an Object

Overview

You can use the COS console to upload objects on the file list page of the bucket.

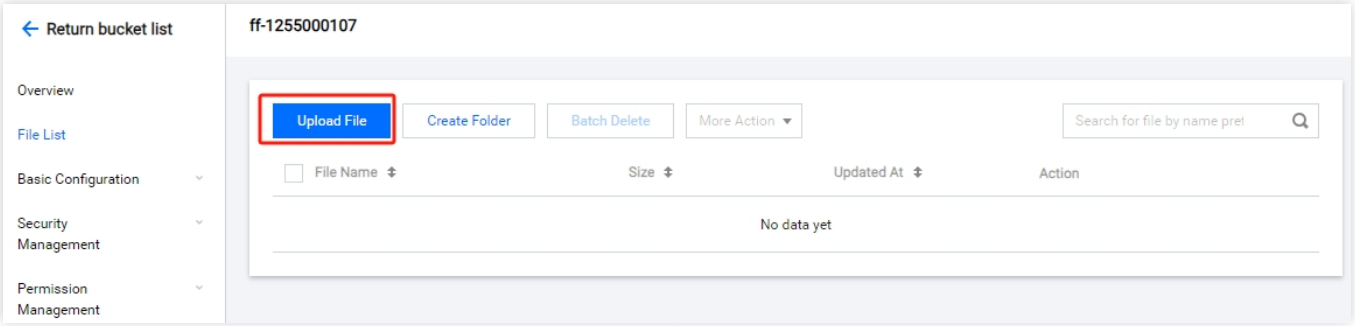
Prerequisites

Before uploading an object, ensure that you have created a bucket.

Directions

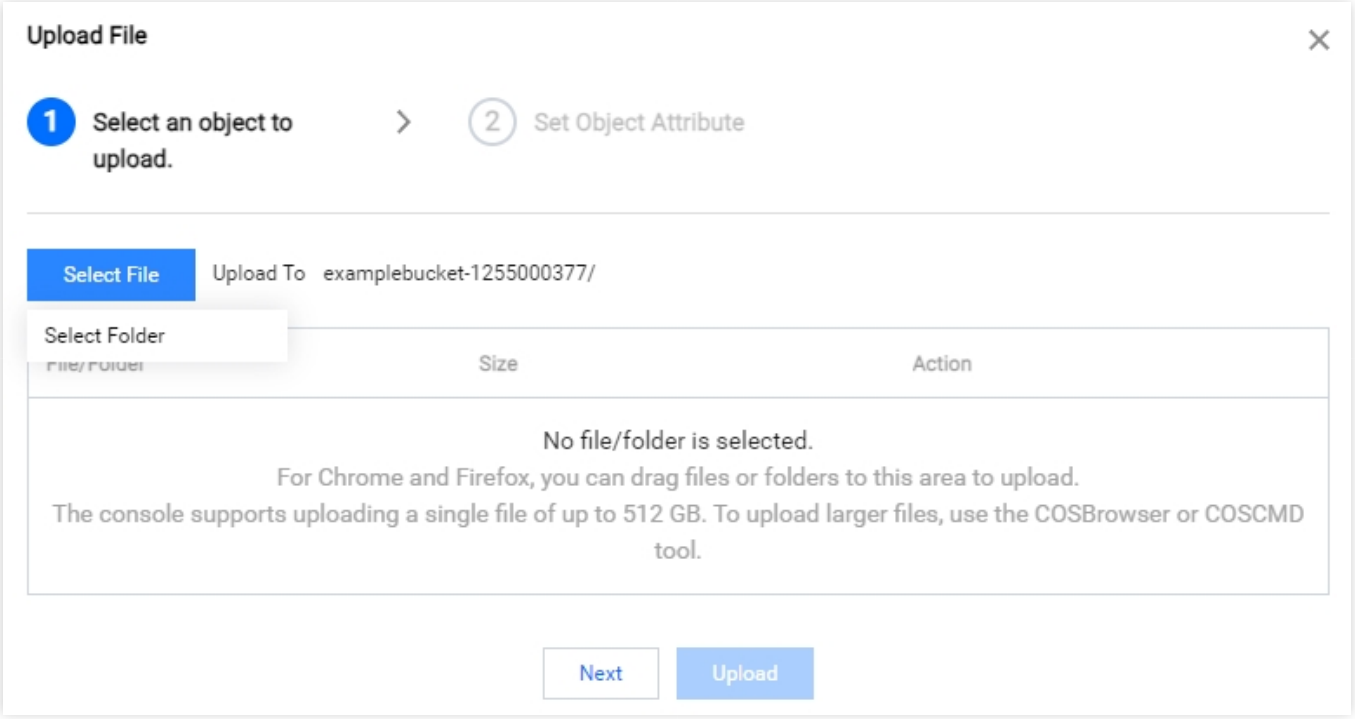
1. Entering the File List

1. Log in to the COS console.
2. In the left navigation tree, click Bucket List to go to the bucket list.
3. Select the bucket where you want to store objects and go to the file list page.
4. Select File List, and click Upload File, as shown in the figure below.



2. Selecting the Upload Object

1. In the Upload File dialog box, click Select File or Select Folder to upload single or multiple local files/ folders.

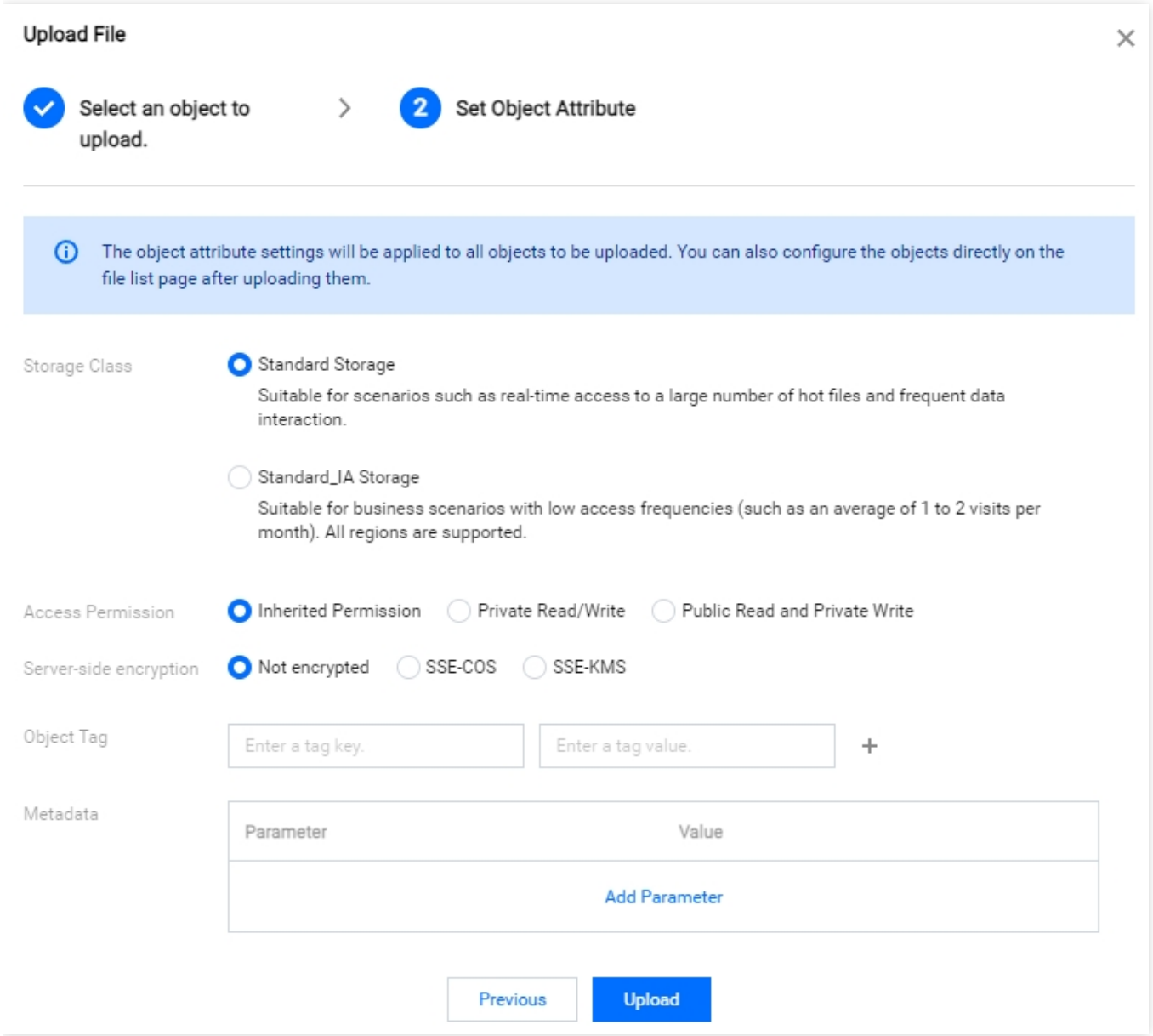


2. Select the local object you want to upload.
3. Click Upload to start uploading the file quickly, or click Next to set object attributes.

3. Setting Object Attributes (Optional)

Set access permission and metadata for the file to be uploaded, and click Upload. The description of the configurations is shown below:

- Storage type: You can set different storage types for objects according to your business scenarios. The default storage type is STANDARD.
- Access permission: You can set different access permissions for objects. The default access permission is Inherit (inheriting permissions of the bucket).
- Object tag: An object tag consists of a tag key (tagKey), an equal sign "=", and a tag value (tagValue). For example, group = IT. You can set, query, and delete tags for specified objects.
- Metadata: Metadata is a string sent by the server before it transmits the HTML data to the browser over the HTTP protocol. Metadata is also called an HTTP header. By modifying HTTP headers, you can change the response form of the page or pass on configuration information, for example, to modify the caching time. Modifying the HTTP headers of an object will not modify the object itself.



> Note:
>
> Some browsers do not support uploading multiple files. It is recommended that you use mainstream browsers, such as Internet Explorer 10 or later, Firefox, and Chrome.

4. Confirming that the Upload is Completed

1. Click Upload. You can check the current upload progress in Task Completed in the top-right corner.
2. After the upload is completed, you will see the uploaded objects in File List.

Downloading Objects

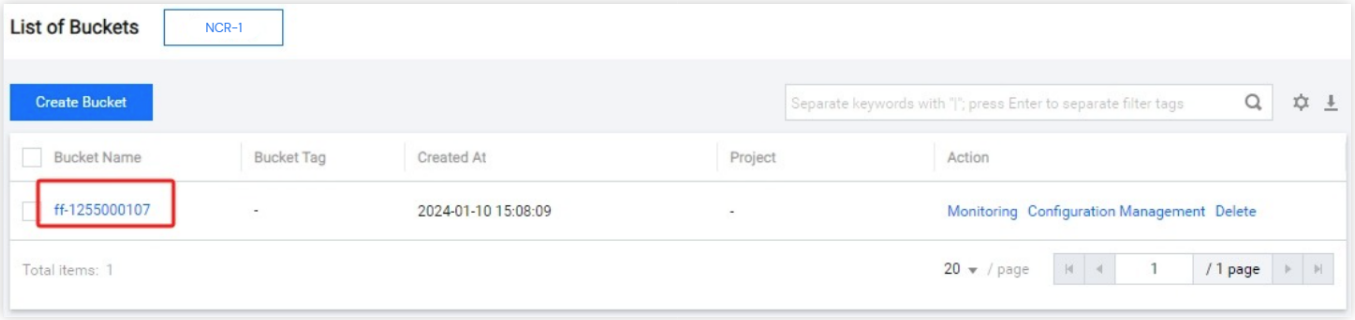
Overview

You can download or access objects that have been uploaded to the bucket through the access address.

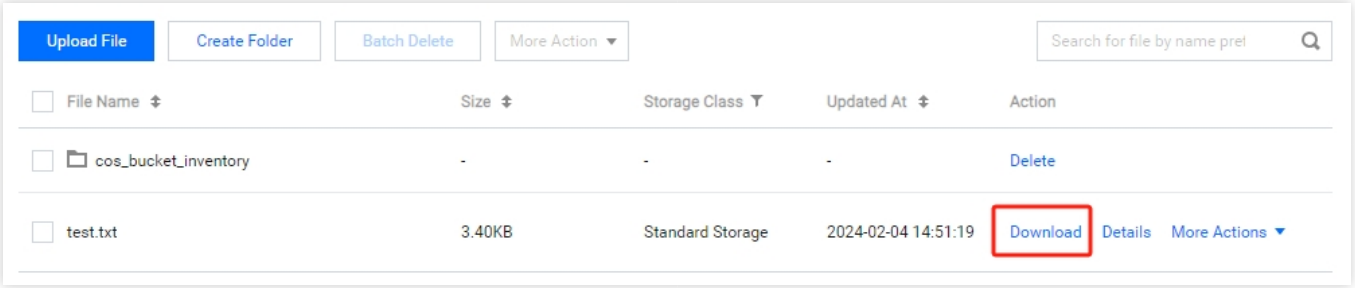
> Note:
>
> Currently, COS only supports downloading a single object but not batch downloading of objects.

Directions

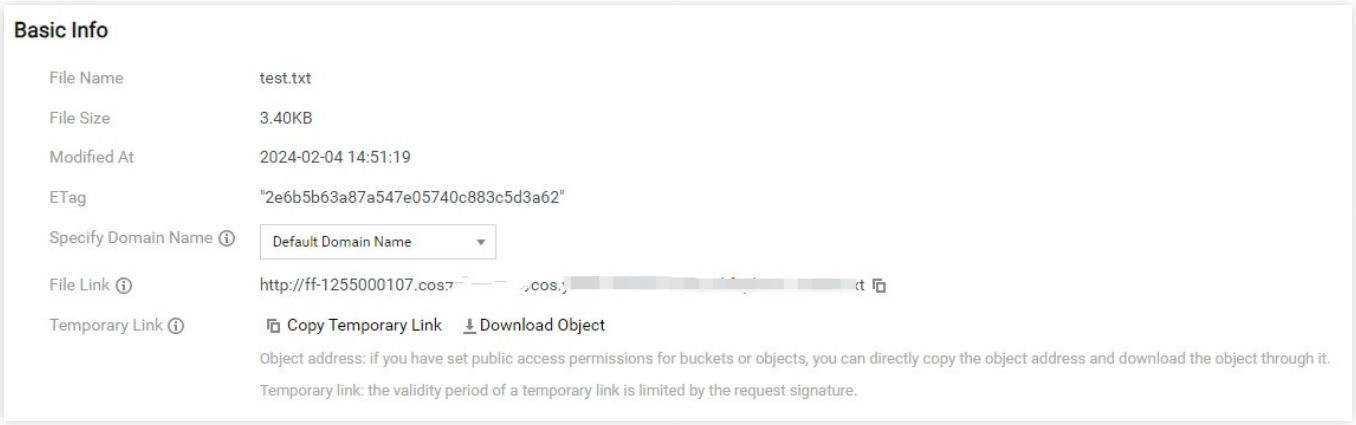
1. Log in to the COS console, click Bucket List, and click the corresponding bucket name to enter the file list of the bucket.



2. Find the object you want to download in the file list and click Download to download the object or click Details on the right to jump to the file details interface to obtain the download link (see step 3).



3. In the file information details box, you can view the object address. You can click Download Object to download directly or click Copy Temporary Link to copy the link, paste it into the browser address column, and press Enter to download.



> Note:
>
> - If the attributes of the bucket to which the object belongs are private read/write, the address copied here will automatically calculate the signature and add a suffix.

Copying Objects

Overview

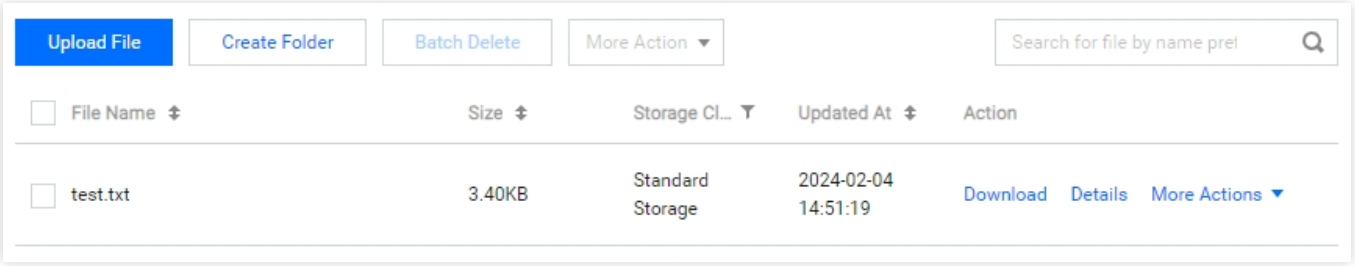
You can use the COS console to copy one or more objects uploaded to a bucket to copy objects from the source path to the target path.

Directions

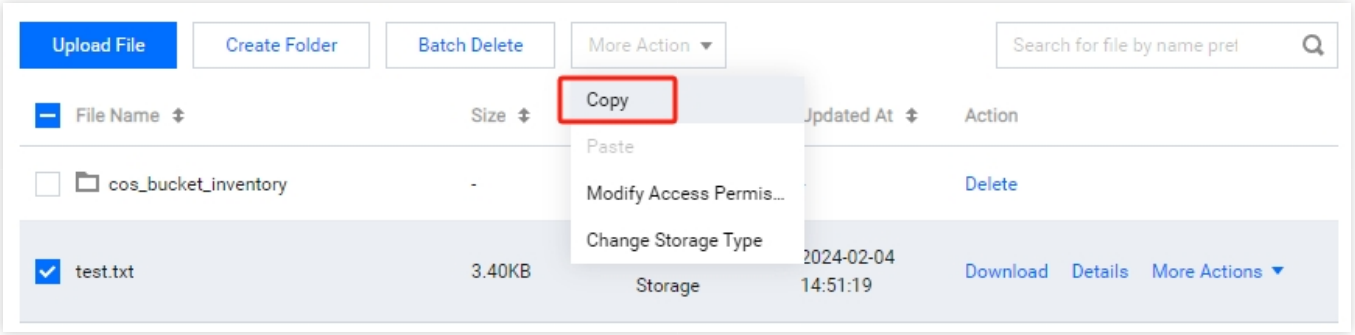
1. Log in to the COS console.

2. In the left navigation tree, click Bucket List to enter.

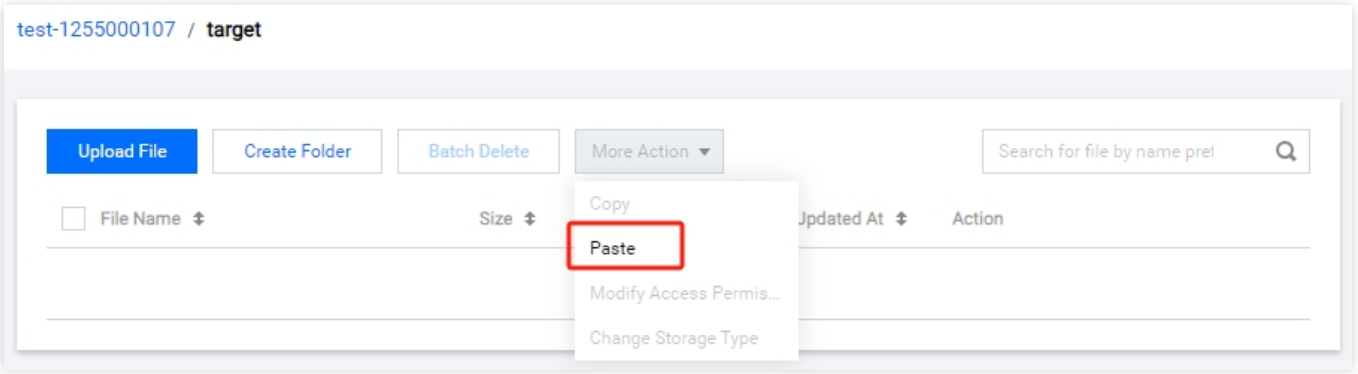
3. Find the corresponding bucket and click the bucket name to enter the file list page.



4. Select the object or folder you want to copy. Multiple selections are supported. Click More Action > Copy.

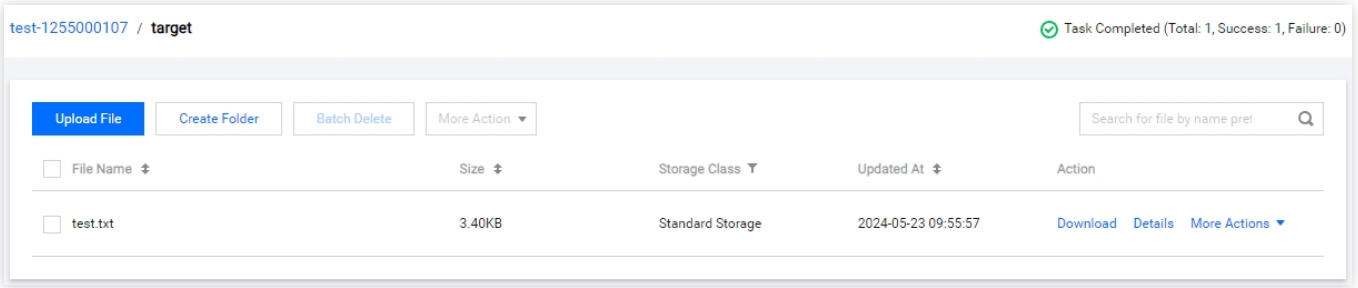


5. After you are prompted that the replication is successful, select the target path and paste the object. For example, you can paste the object to the target folder in the bucket test-1255000107.



> Note:
>
> The target path cannot be the same as the source path. Otherwise, the paste will fail.

6. After pasting successfully, you can see that the objects or folders are copied to the target folder.



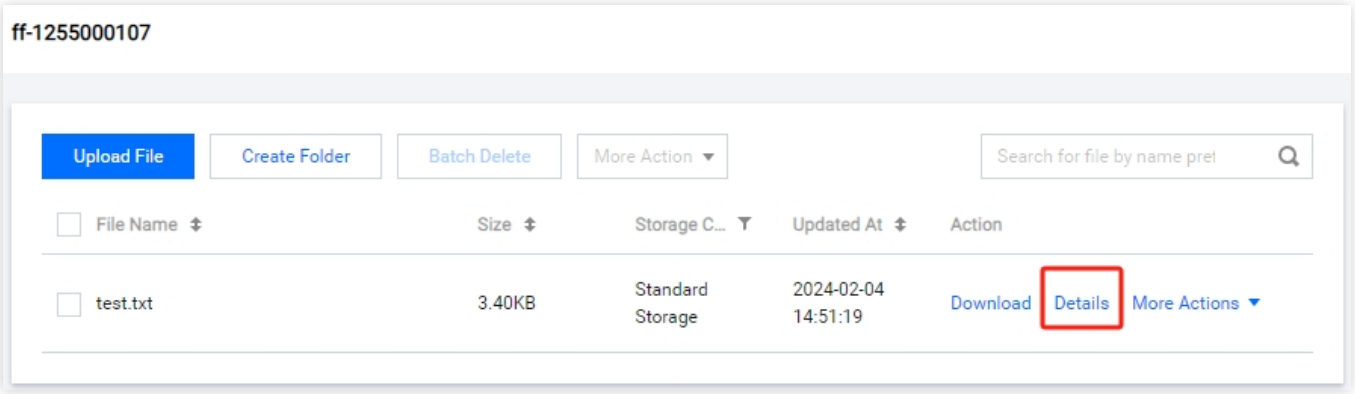
Viewing Object Information

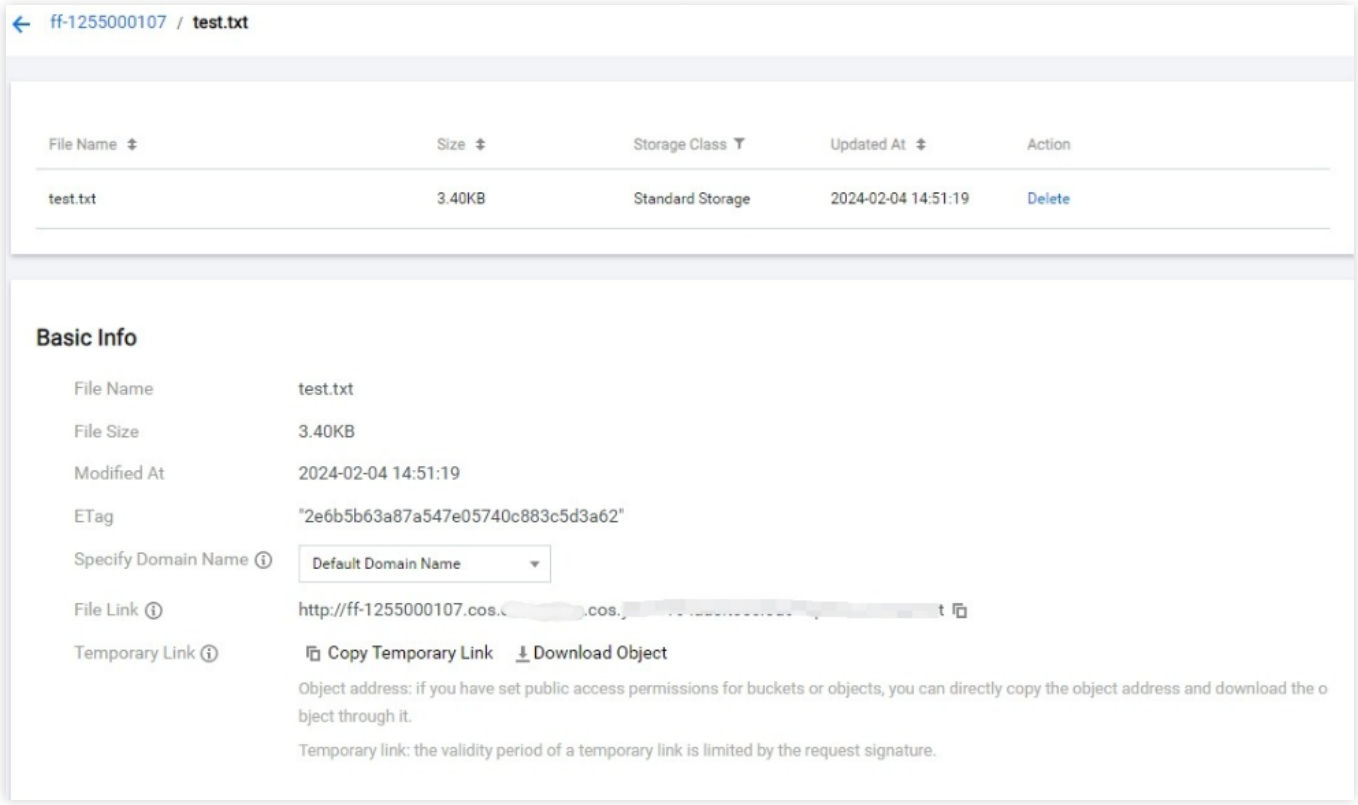
Overview

The COS console lets you view object attribute information (such as object size and object address) and object-related configurations (such as setting object access permissions, changing storage types, etc.).

Directions

1. Log in to the COS console and click Bucket List to enter.
2. Click the name of the bucket where the object is located to enter the File List page of the bucket.
3. Click Details on the object's right side to view the object size and object address and obtain the signature link and other related information. At the same time, you can also configure the object.




ff-1255000107 / test.txt
File Name
Size
Storage Class
Updated At
Action
test.txt
3.40KB
Standard Storage
2024-02-04 14:51:19
Delete
Basic Info
File Name
test.txt
File Size
3.40KB
Modified At
2024-02-04 14:51:19
ETag
"2e6b5b63a87a547e05740c883c5d3a62"
Specify Domain Name
Default Domain Name
File Link
http://ff-1255000107.cos.
.cos.
Temporary Link
Copy Temporary Link
Download Object
Object address: if you have set public access permissions for buckets or objects, you can directly copy the object address and download the object through it.
Temporary link: the validity period of a temporary link is limited by the request signature.

Searching for Objects

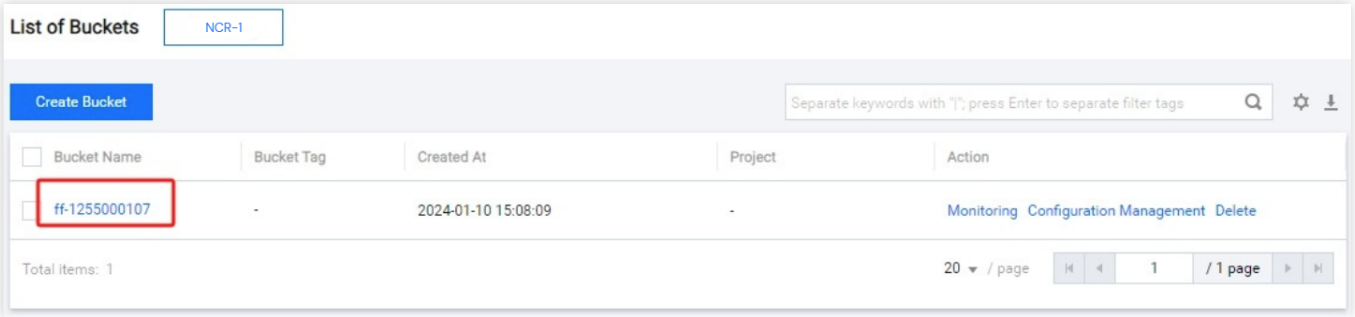
Overview

You can use the COS console to search for the uploaded objects.

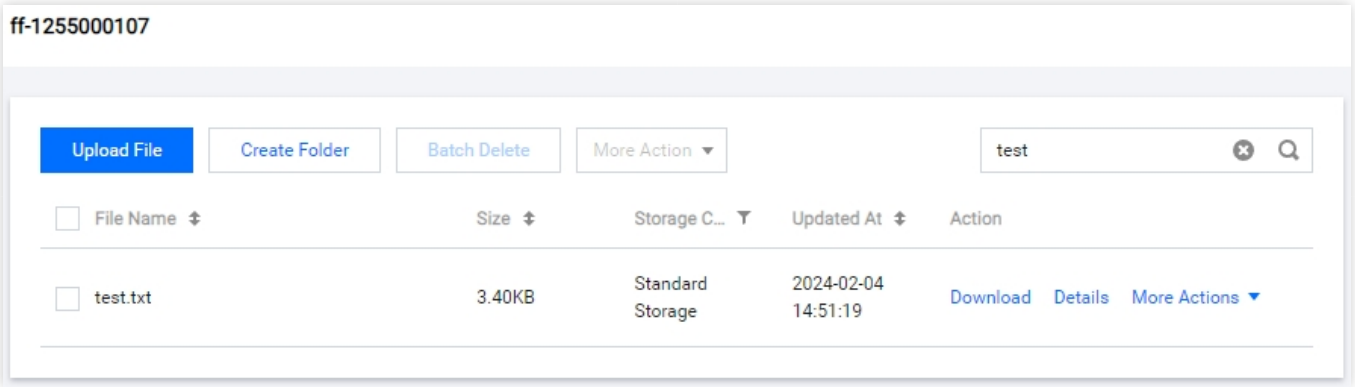
Searching for the Current Objects in the Bucket

Directions

1. Log in to the COS console, click Bucket List to go to the bucket list, and click the bucket where the object is located.



2. In the search box in the top-right corner of File List, type the object name prefix, and click the search button. Then, the objects or folders with the same name prefix in the current bucket are displayed. If you need to search for a specific object, you can type the complete object key, such as test.txt .



Sorting or Filtering Objects

Overview

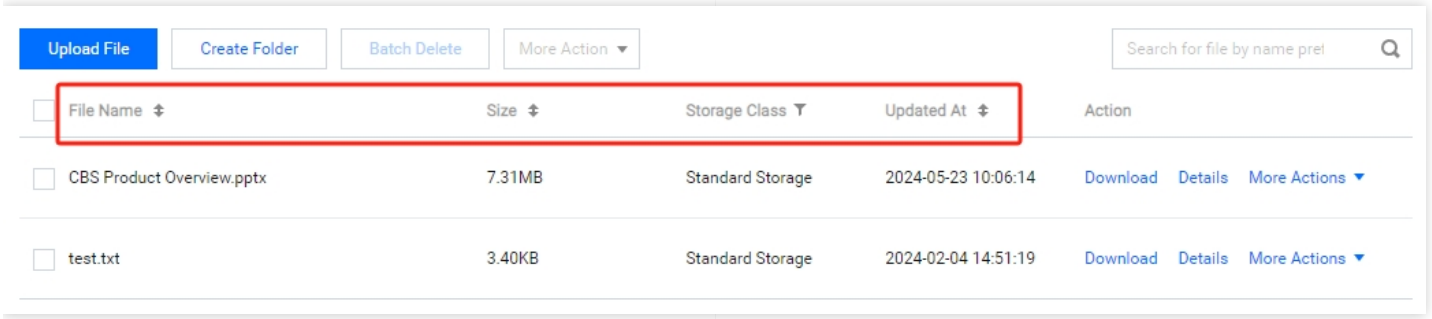
You can sort and filter files in the bucket file list through the COS console.

> Note:
>
> The sorting and filtering operations can be performed only if the number of files and folders is less than 1,000.

Directions

1. Log in to the COS console.
2. In the left sidebar, select Bucket List to go to the bucket list page.
3. Find the corresponding bucket and click the bucket name to enter the file list page.
4. Click the table header to sort or filter according to actual needs.

   > Note:
   >
   > Currently supports sorting the file list by file name, size, modification time, and filtering by storage type.



Renaming a File

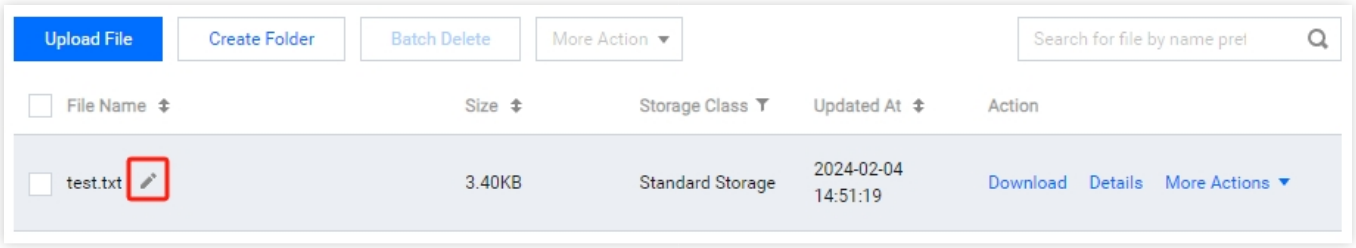
Overview

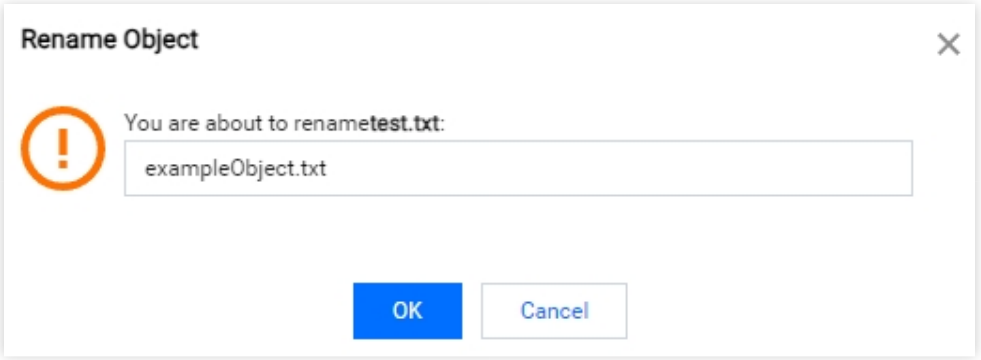
You can use the COS console to rename the uploaded objects.

Directions

1. Log in to the COS console.
2. In the left sidebar, select Bucket List to go to the bucket list page.
3. Click the corresponding bucket name to go to the bucket file list.
4. Move the cursor over the name of an uploaded object and click the edit button.

> Note:
>
> The name cannot start with "/" or "****", and its length should be up to 850 characters.





Modifying Storage Type

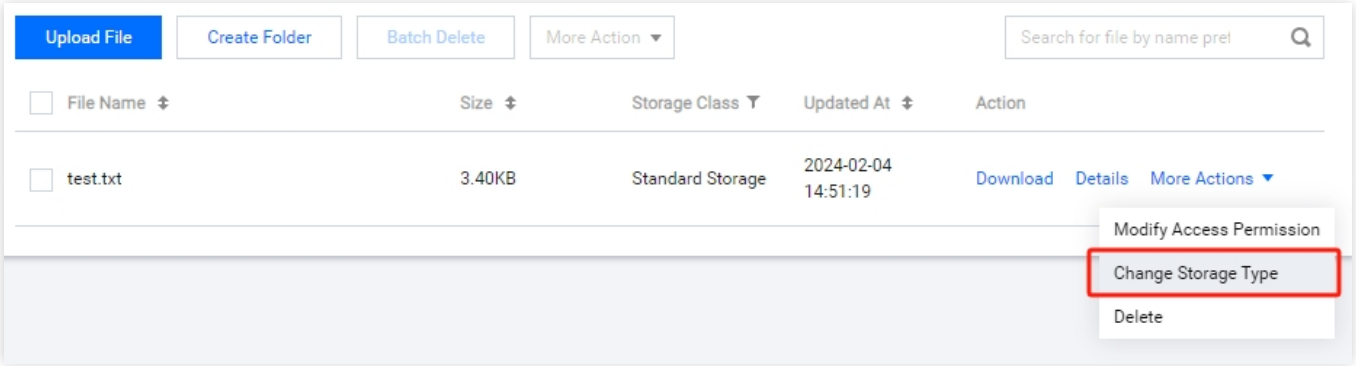
Overview

You can modify the storage type of uploaded objects at any time through the COS console to meet the business needs of different scenes.

> Note:
>
> The storage type of an object larger than 5 GB cannot be modified.

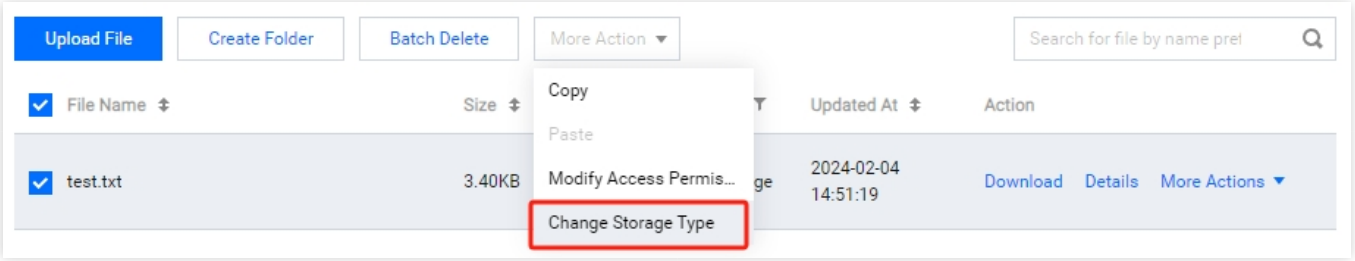
Directions

1. Log in to the COS console.
2. In the left sidebar, click Bucket List to go to the bucket list page.
3. Find the bucket where the object is located and click the bucket name to enter the file list page.
4. Find the object whose storage type you want to modify. In the operation column on its right, click More Actions > Change Storage Type to set it.



If you need to modify the storage type of multiple objects in batches, you can check multiple objects and click More Action > Change Storage Type at the top to modify it.

> Note:
>
> Only batch modification of the same storage type is supported.


Upload File
Create Folder
Batch Delete
More Action
Search for file by name pref
File Name
Size
Updated At
Action
Copy
Paste
Modify Access Permis...
Change Storage Type
test.txt
3.40KB
2024-02-04
14:51:19
Download
Details
More Actions

Setting Access Permissions for an Object

Overview

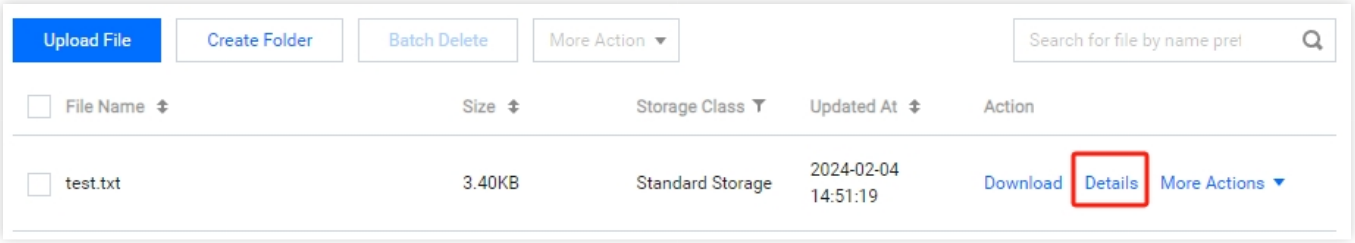
COS allows you to set access permissions based on the object dimension, and such permissions have a higher priority than those for buckets. You can set the following two types of permissions for objects in COS:

- Public Permission: Includes Inherit, Private Read/Write, and Public Read/Private Write.
- User Permission: The root account has all permissions (full control) on the objects by default. In COS, you can also add sub-accounts and grant them permissions, including read/write data, read/write permissions, and even Full Control, namely the highest permission level.

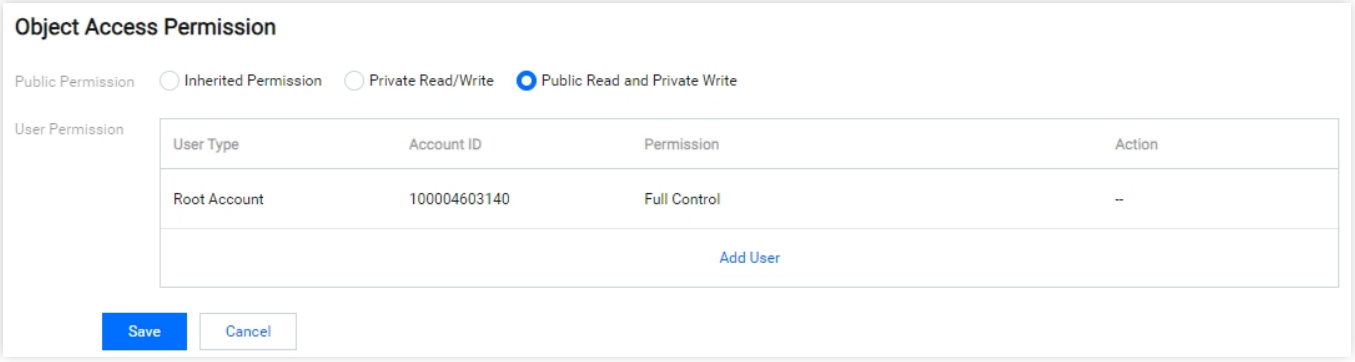
> Note:
>
> The access permissions of the object are only valid when the user accesses it through the default domain name. Subject to the bucket access permission when accessing through a custom domain name.

Directions

1. Log in to the COS console.
2. In the left sidebar, click Bucket List, and click the name of the bucket where the object is located to go to the bucket file list page.
3. Find the object you want to set permissions and click Details on the right.



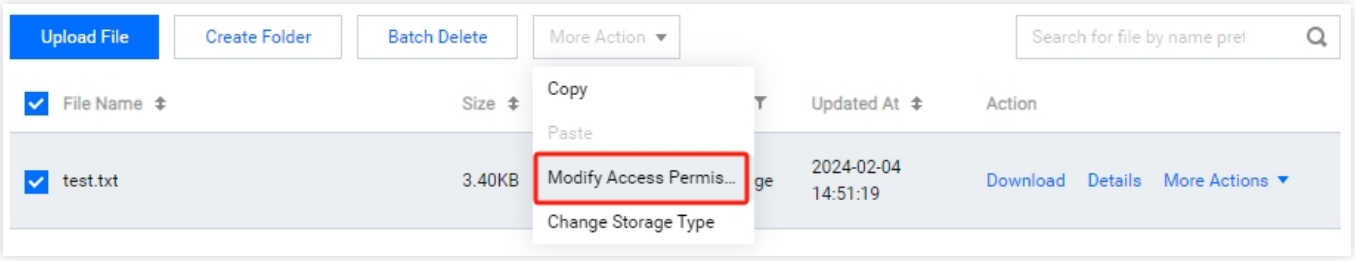
4. In Object Access Permission configuration, after modifying the access permissions, click Save.



The COS supports two types of permissions for objects:

- Public permissions: Include Inherit, Private Read/Write, and Public Read/Private Write.
- User permissions: The root account has all permissions (full control) on the objects by default. In COS, you can also add sub-accounts and grant them permissions, including read/write data, read/write permissions, and even full control, namely the highest permission level.

5. If you need to set or modify access permissions for multiple objects in batches, you can check multiple objects, and click More Action > Modify Access Permission at the top to set it.



Setting Object Encryption

Overview

You can use the COS console to encrypt the objects stored in buckets to prevent data leakage.

> 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.
> - When you list objects in a bucket, all objects will be listed, regardless of whether they are encrypted.

Operation Steps

1. Log in to the COS console.
2. In the left sidebar, click Bucket List, and go to the bucket list page.
3. Find the bucket where the object is located and click the bucket name to enter the bucket management page.
4. In the left sidebar, select File List to go to the file list page.
5. Find the object you want to set encryption on and click Details on the right operation column.
6. In the Server-Side Encryption area, click Modify.
7. Select the corresponding encryption method and click Save.
   Currently, the following two encryption methods are supported:
   - SSE-COS: Server-side encryption with COS-managed keys.
   - SSE-KMS: Server-side encryption with a key managed by KMS. You can choose to use the default key or create a key.

     > Note:
     >
     > SSE-C encryption is server-side encryption with a custom key. The encryption key is provided by the user. When the user uploads the object, COS will use the encryption key provided by the user to encrypt the user's data by AES-256. (SSE-C can only be used via API and does not support console operations)

Example: Using Server Encryption SSE-C

Request

```
PUT /exampleobject HTTP/1.1
Host: <bucketname-appid>.<Endpoint>
Date: Fri, 10 Apr 2020 09:36:12 GMT
Content-Type: image/jpeg
x-cos-server-side-encryption-customer-algorithm: AES256
x-cos-server-side-encryption-customer-key: MDEyMzQ1Njc4OUFCQ0RFRjAxMjM0NTY3ODlBQkNERUY
=
x-cos-server-side-encryption-customer-key-MD5: U5L61r7jcwdNvT7frmUG8g==
Content-Length: 16
Content-MD5: 7o3pGNBWQBRbGPcPTDqmAg==
Authorization: q-sign-algorithm=sha1&q-ak=AKID8A0fBVtYFrNm02oY1g1JQQF0c3JO****&q-sign-tim
e=1586511372;1586518572&q-key-time=1586511372;1586518572&q-header-list=content-length;con
tent-md5;content-type;date;host;x-cos-server-side-encryption-customer-algorithm;x-cos-server-side-e
ncryption-customer-key;x-cos-server-side-encryption-customer-key-md5&q-url-param-list=&q-signat
ure=4f6f9f0a6700930f70bff31e3a2b2e622711****
Connection: close
```

Response

```
HTTP/1.1 200 OK
Content-Length: 0
Connection: close
Date: Fri, 10 Apr 2020 09:36:13 GMT
ETag: "582d9105f71525f3c161984bc005efb5"
x-cos-hash-crc64ecma: 16749565679157681890
x-cos-request-id: NWU5MDNlMGNfZTFjODJhMDlfMzVlMDFfZTk1****
x-cos-server-side-encryption-customer-algorithm: AES256
x-cos-server-side-encryption-customer-key-MD5: U5L61r7jcwdNvT7frmUG8g==
```

Custom Headers

Overview

The HTTP header of an object is a string sent by the server before it transmits the HTML data to the browser over the HTTP protocol. By modifying HTTP headers, you can change the response form of the page or pass on configuration information, for example, to modify the caching time. Modifying the HTTP headers of an object will not modify the object itself.
For example, if the Content-Encoding in the header is modified to gzip, but the file itself has not been compressed with gzip in advance, then a decode error will occur.

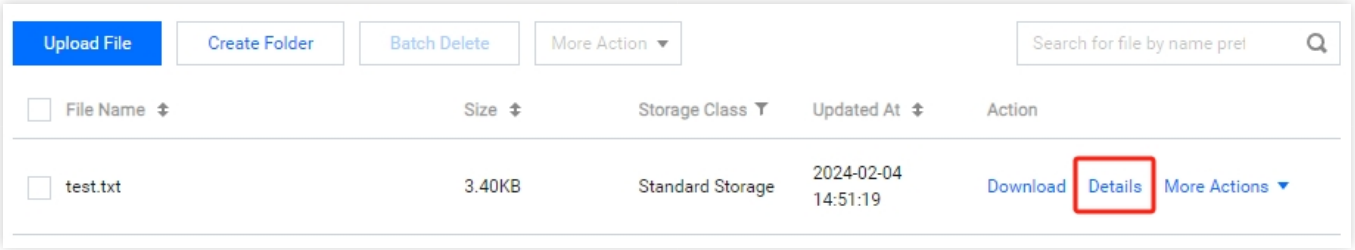
Operation Steps

1. Log in to the COS bucket console.

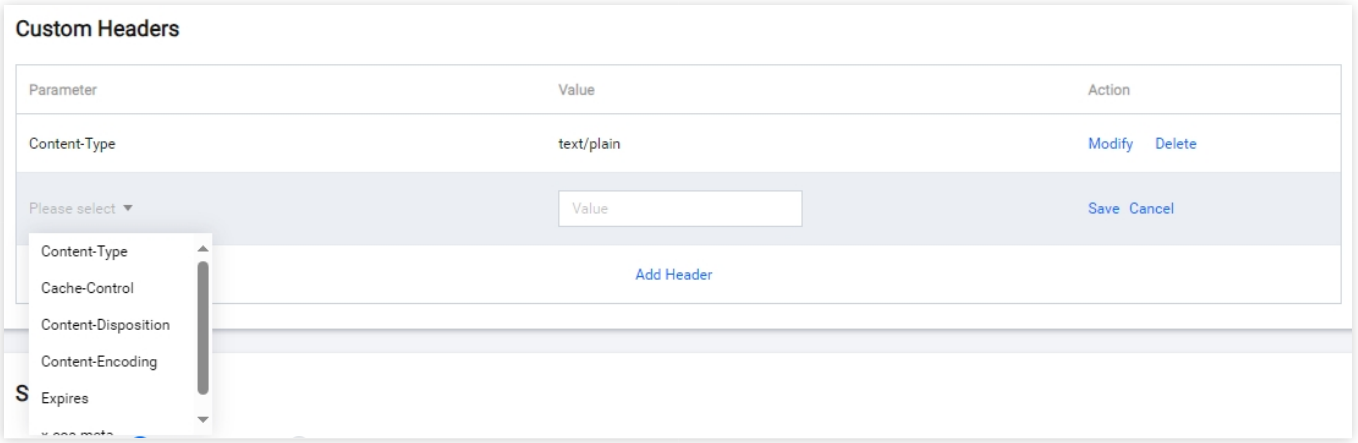
2. In the left sidebar, select Bucket List to go to the bucket list page.

3. Click the bucket where the object is located to enter the bucket file list.

4. Find the object you need to set the header for and click Details on the right side of the object.



5. In the Custom Headers configuration, click Add Header, select the parameter type you need to set (for Custom Content, enter a custom name), and enter the corresponding value.



COS offers the following 6 types of object HTTP header identifiers for configuration. The header configuration is as follows.

HTTP Header	Description	Example
Content-Type	MIME information of the file.	image/jpeg
Cache-Control	Caching mechanism of the file.	no-cache;max-age=200
Content-Disposition	Extension of the MIME protocol.	attachment;filename="fname.ext"
Content-Encoding	Encoding format of the file.	UTF-8
Expires	To control the expiration date of the cache.	Wed, 21 Oct 2015 07:28:00 GMT
x-cos-meta-[Custom Content]	Custom content.	Custom content.

6. After completing the configuration, click Save.

Example

For APPID as 1250000000, create a bucket named examplebucket-1250000000. In the root directory of the bucket, an object exampleobject.txt was uploaded.

When not customizing an object's HTTP headers, the range of object headers obtained by the browser or client during download is as follows:

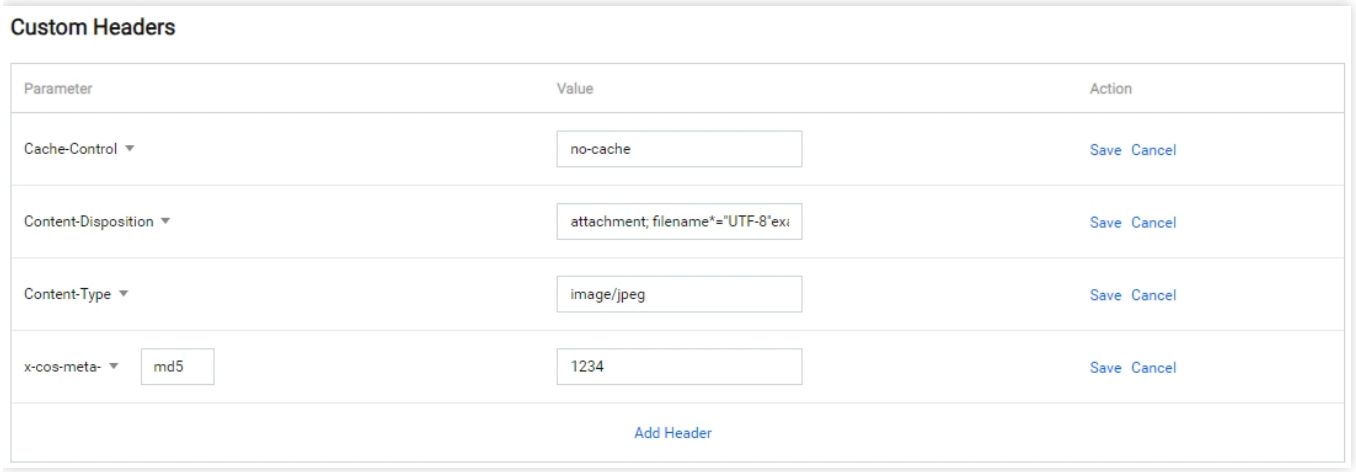
Request

```
GET /exampleobject.txt HTTP/1.1
Host: <bucketname-appid>.<Endpoint>
Accept: */*
```

Response

```
HTTP/1.1 200 OK
Content-Language:zh-CN
Content-Type: text/plain
Content-Disposition: attachment; filename*="UTF-8''exampleobject.txt"
Access-Control-Allow-Origin: *
Last-Modified: Tue, 11 Jul 2017 15:30:35 GMT
```

Add the following configurations:



Initiate the request again, the object header example received by the browser or client is as follows:

Request

```
GET /exampleobject.txt HTTP/1.1
Host: <bucketname-appid>.<Endpoint>
Accept: */*
```

Response

```
HTTP/1.1 200 OK
Content-Language:zh-CN
Cache-Control: no-cache
Content-Type: image/jpeg
Content-Disposition: attachment; filename*="abc.txt"
x-cos-meta-md5: 1234
Access-Control-Allow-Origin: *
Last-Modified: Tue, 11 Jul 2017 15:30:35 GMT
```

Deleting an Object

Overview

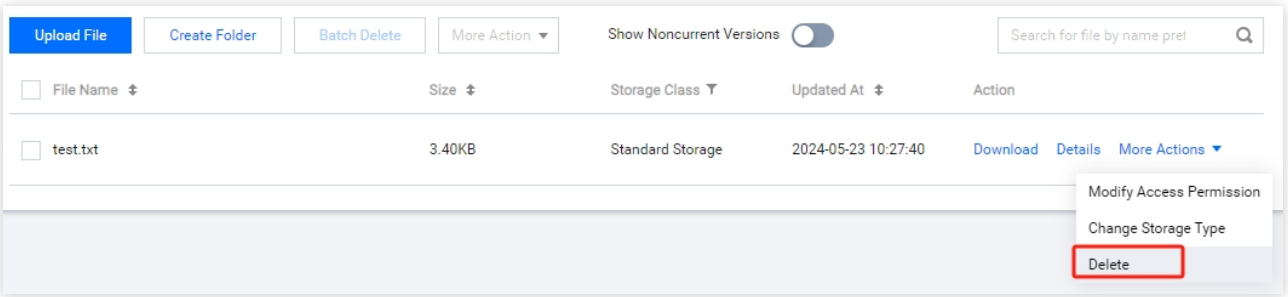
You can use the COS console to delete one or more objects uploaded to a bucket.

Deleting an Object

Directions

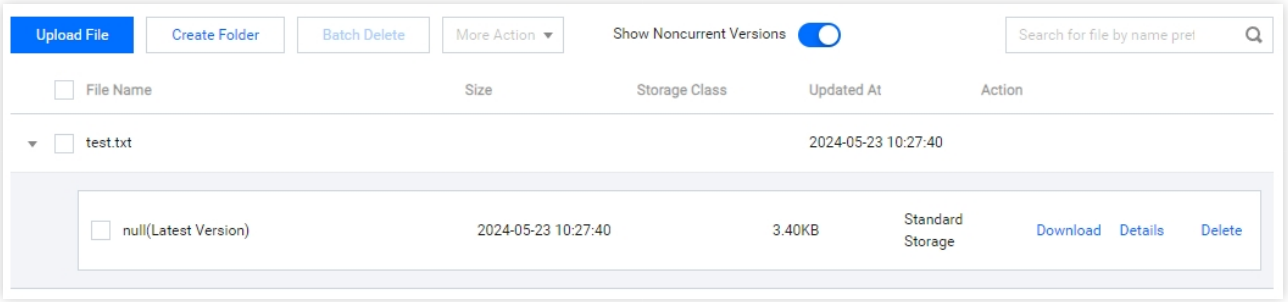
1. Log in to the COS console.
2. Select Bucket List in the left sidebar to go to the bucket list page.
3. Click the bucket where the object is located to enter the bucket file list.
   - No historical versions listed

     Find the object you want to delete and click More Actions > Delete.



   - Historical versions listed

     Find the object you want to delete and click Delete.



4. The delete file dialog box pops up, click Confirm to delete the object.

Deleting Multiple Objects

Directions

1. On the bucket file list page, select multiple objects you want to delete and click Batch Delete.
2. The delete file dialog box pops up. Click Confirm to delete the objects in batch.

Folder Management
Creating a Folder

Resources are stored in the form of objects in COS. To continue user usage habits, objects can be managed in the form of folders in the COS console.

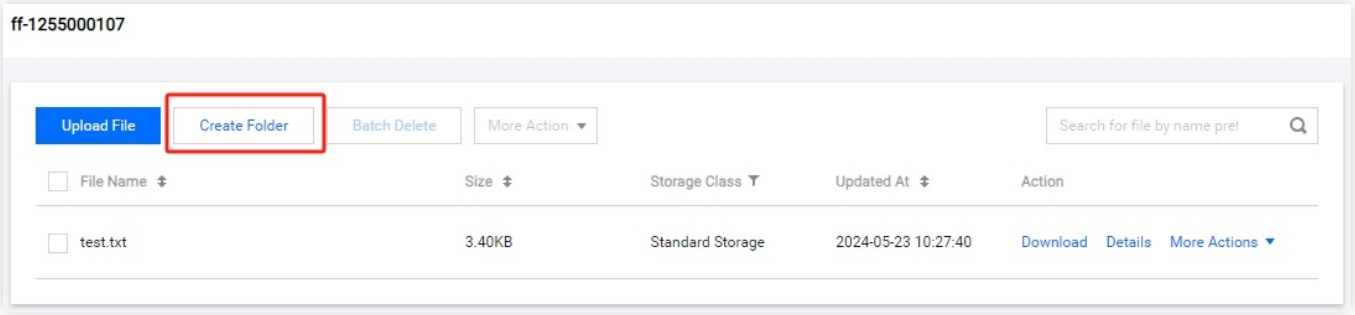
> Note:
>
> Folder names are limited to 255 characters in length, and reserved characters and fields are not supported.

Reserved Characters and Fields

- Reserved fields: [con], [aux], [nul], [prn], [com0], [com1], [com2], [com3], [com4], [com5], [com6], [com7], [com8], [com9], [lpt0], [lpt1], [lpt2], [lpt3], [lpt4], [lpt5], [lpt6], [lpt7], [lpt8], and [lpt9].
- To preserve ASCII control characters:
  Character up (↑): CAN (24)
  Character down (↓): EM (25)
  Character right (→): SUB (26)
  Character left (←): ESC (27)

Directions

1. Log in to the COS bucket console.
2. In the left sidebar, select Bucket List, and go to the bucket list page.
3. Locate the bucket for which you want to create a folder, and click its name to enter the bucket list.
4. Click Create Folder.

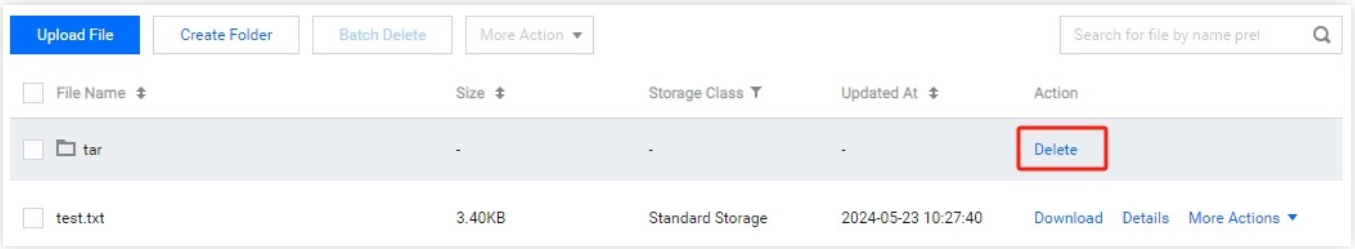


5. Input the folder name and then click OK.

Deleting a Folder

Directions

1. Log in to the COS bucket console.
2. In the left sidebar, select Bucket List, and go to the bucket list page.
3. Locate the bucket for which you want to delete a folder, and click its name to enter the bucket list.
4. Locate the folder to be deleted and click Delete.



5. Click Confirm to delete the folder and all files in the directory.

Setting Object Tags

Overview

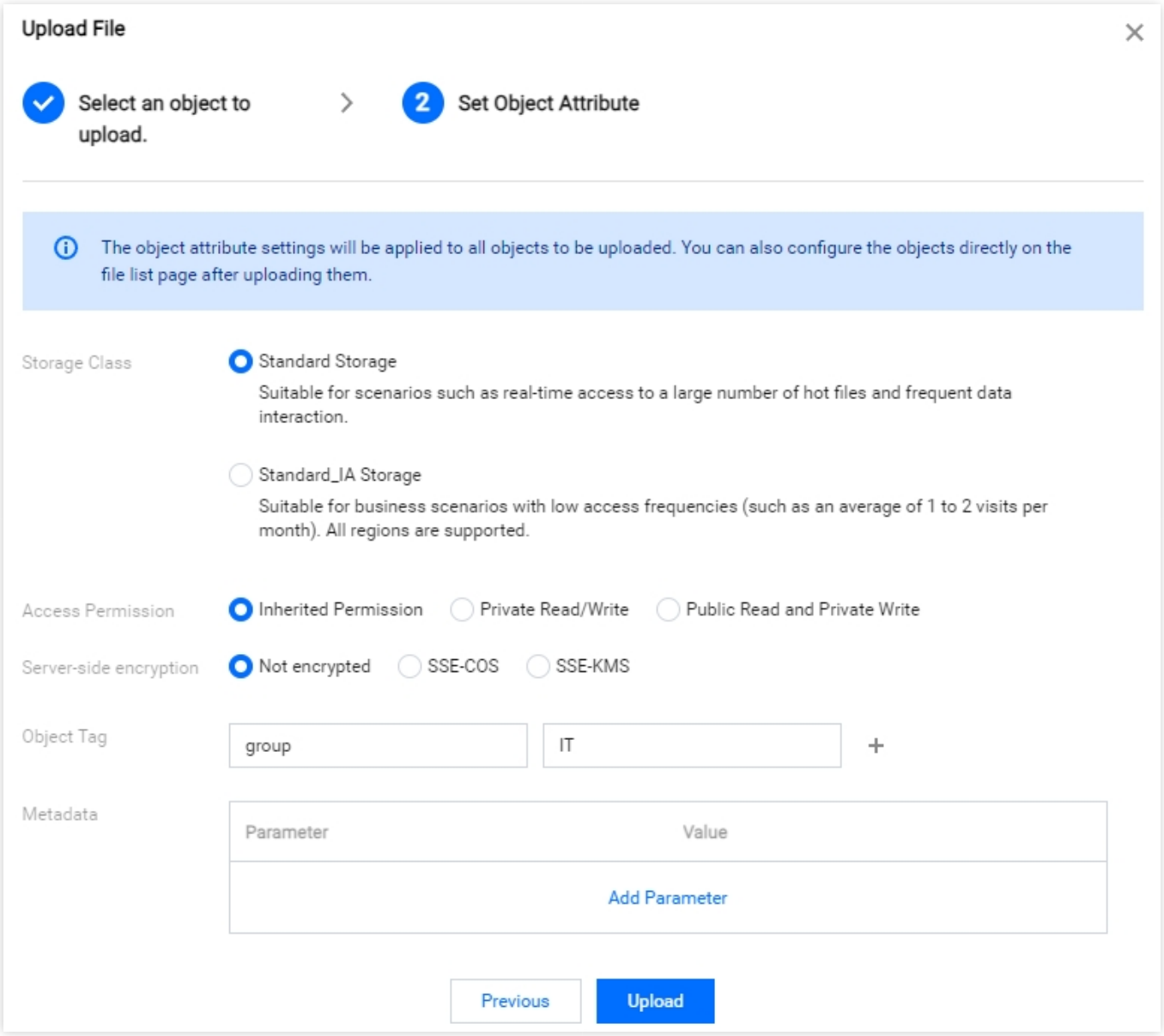
The implementation of the object tag feature is to assist users in grouping and managing objects in the storage bucket by adding a key-value pair form of identification to the object. The object tag is composed of the tagKey and the tagValue connected to=, for example, group = IT. Users can set, query, and delete tags for specified objects.

You need to pay attention to the following use limits of tags:

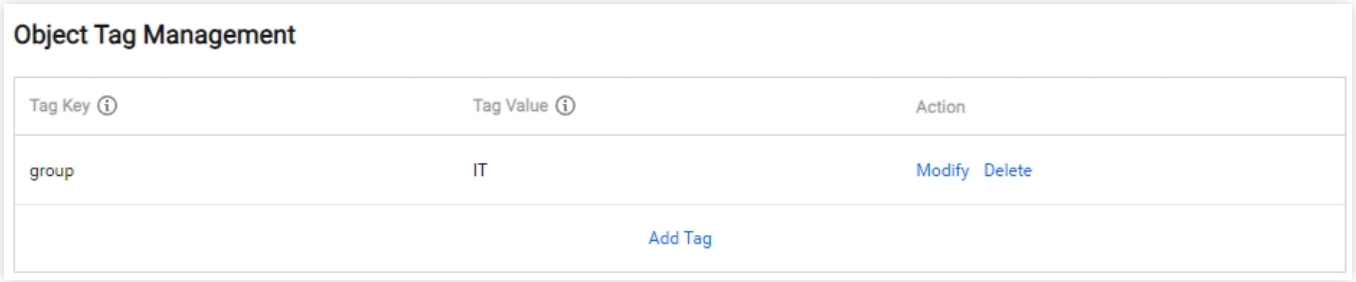
- Users can add up to 10 object tags for the same object, and tags cannot be repeated.
- The tag keys are not recommended to begin with qcs: or project. These characters are reserved for system tag keys.
- Tag keys and tag values support characters in UTF-8 format, spaces, numbers, and special characters + - = ._ : / @. The length range is 1-127 characters and they are case sensitive.

Adding Tags When Uploading Objects

1. You can add it when uploading an object, as shown in the following figure:



2. After the upload is successful, the object tag is added.
   You can go to Object Tag Management in the object details page to view the added tags:

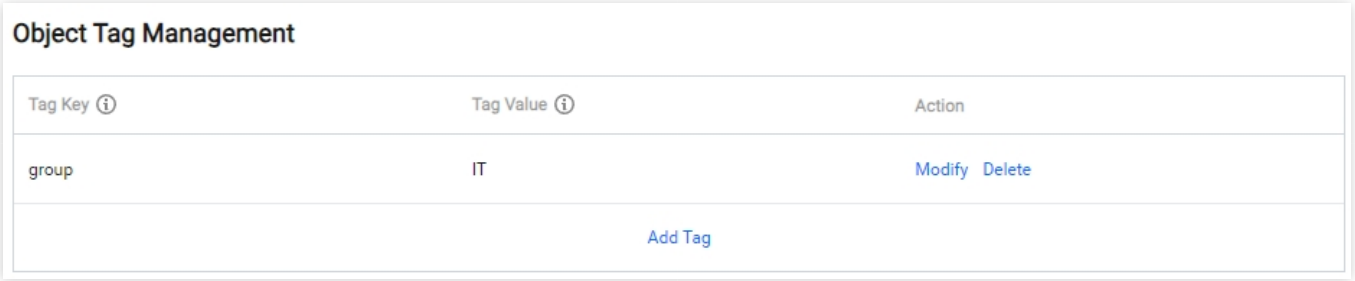


3. If you need to modify or delete a tag, you can go to the configuration item of Object Tag Management and click Modify or Delete on the right side of the tag.

Adding Tags to Uploaded Objects

If you did not add tags when you uploaded the object, you can add tags to the object by following the steps below.

1. See View Object Information to enter the object details page where you want to add tags.
2. In the Object Tag Management configuration item on the object details page, click Add Tag to add a tag to the object.



Adding Tags to Existing Objects

If you need to modify or delete a tag, you can go to the configuration item of Object Tag Management and click Modify or Delete on the right side of the tag.

Restoring Historical Version Objects

Overview

You can restore the historical version of an object to the latest version through the COS console. This document introduces how to restore a historical version of a bucket object in the console.

> Note:
>
> - Restoring an object means restoring the historical version to the latest version while still retaining the historical version.
> - If you need to delete historical versions, see Setting Versioning and Deleting an Object operations.
> - The operation supports single-object restoration but does not support batch restoration.
> - The restoration is not supported for encrypted objects.

Note

The applicable scenarios and rules for restoring objects are as follows:

Applicable Scenario	• Buckets with versioning enabled support object restoration. • For a bucket that has versioning enabled and then paused, the object restoration is not supported. • Buckets that have not enabled versioning do not support object restoration.
Restoration Rules	• Historical version: Support restoration. • Latest version: Cannot be restored. • Version with deletion marker: Cannot be restored.

Prerequisites

Before restoring an object, make sure versioning is enabled for the bucket. If it is not enabled, see the documentation Setting Versioning.

Directions

1. Log in to the COS console.
2. In the left sidebar, click Bucket List to go to the bucket list page.
3. Find the bucket where the object is located and click the bucket name to go to the file list page.
4. Turn on the Show Noncurrent Versions switch, find the object you want to restore, and click Restoration.
5. In the Restore File dialog box, check the version information of the restored object. After confirmation, click OK.
   After the object restoration is completed, you can see in the file list that the historical version has been successfully restored.

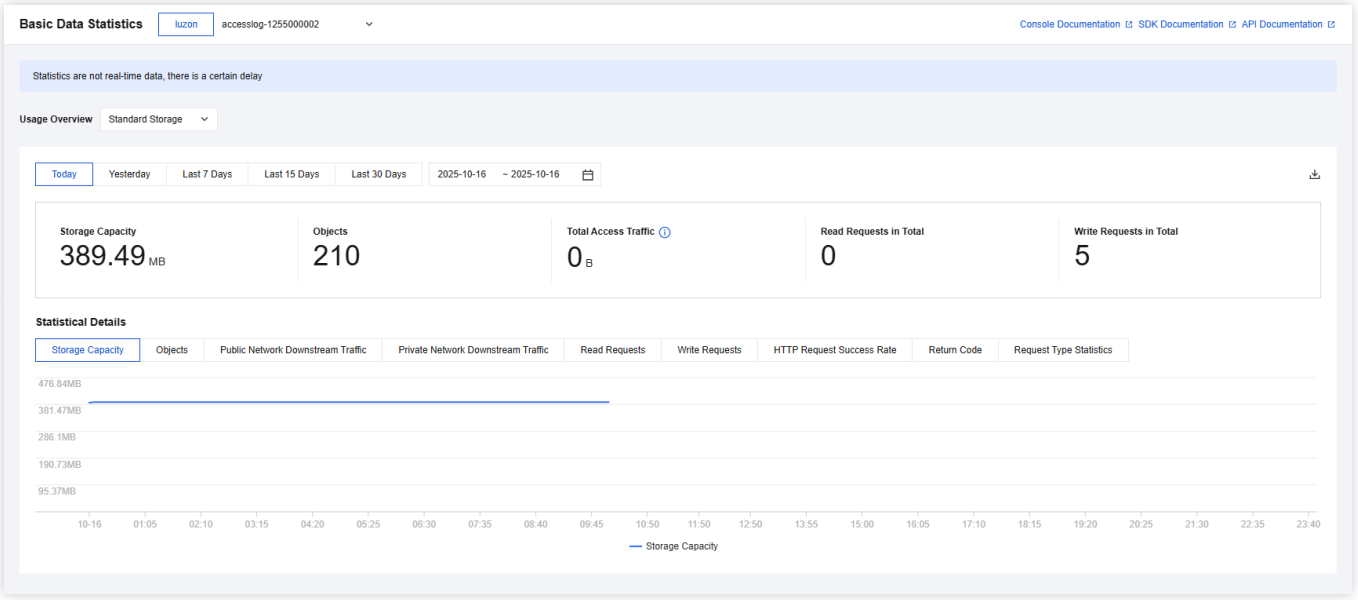
Data Statistics
Basic Data Statistics

Overview

Monitoring reports provide users with service data statistics for object storage. Users can understand the trends of various data through monitoring report data. Monitoring reports include basic data statistics and return code statistics. The basic data statistics page provides users with specific service statistics and graphically displays the service usage in the form of trend charts.

Directions

1. Log in to the COS console.
2. Click Monitoring Report > Basic Data Statistics on the left menu to enter the basic data statistics page.

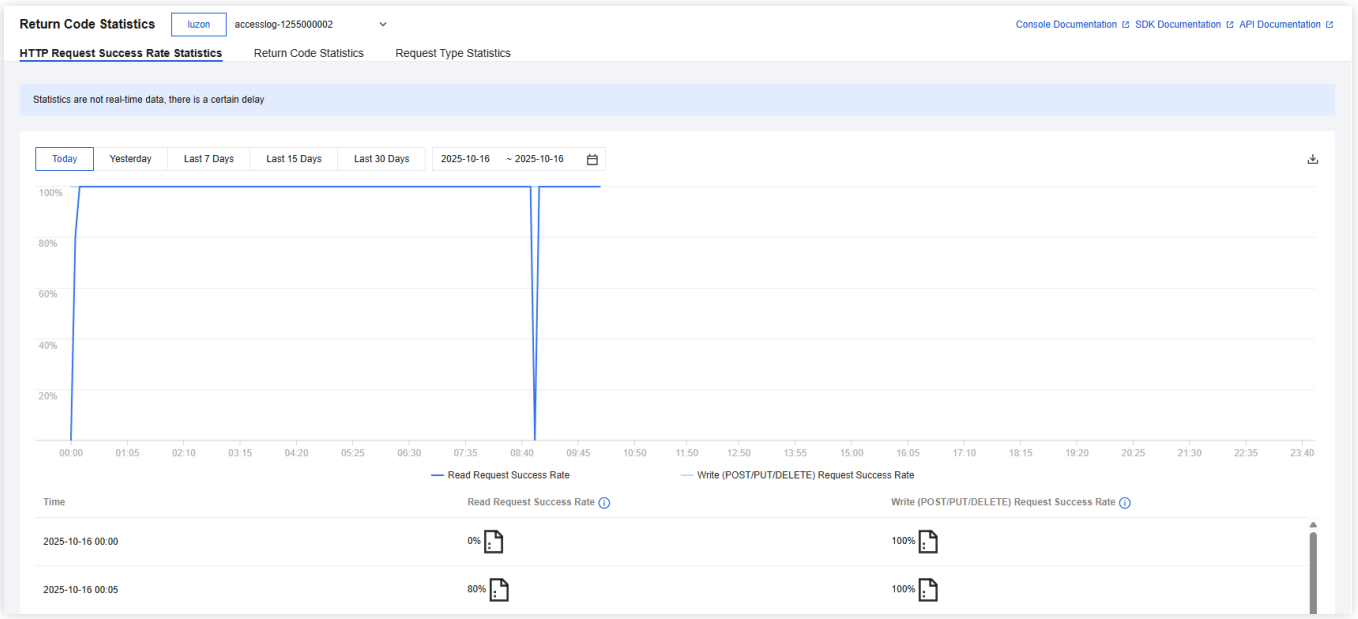


- The basic data statistics page allows you to switch regions, buckets, and storage types to view data for today, yesterday, the past 7 days, the past 15 days, and the past 30 days:
  - The corresponding period is displayed with specific statistics: storage capacity, number of objects, total access traffic, total number of read requests, and total number of write requests.
  - The corresponding period is displayed in a trend chart: storage statistics, quantities of objects, public network downstream traffic, private network traffic, read request statistics, and write request statistics.
- The page supports exporting data to local storage. Click the export button on the right side of the date column to export data. There may be a delay in data, so refresh the page to reload it.

Return Code Statistics

Directions

1. Log in to the COS console.
2. Click Monitoring Report > Return Code Statistics on the left menu to go to the return code statistics page.



- The return code statistics page is in the form of trend charts and lists for the corresponding periods. You can switch projects and buckets to view data for the current day, yesterday, the past 7 days, the past 15 days, and the past 30 days:
  - HTTP request success rate statistics
    The HTTP request success rate is calculated by the ratio of the number of occurrences of each return code to the total number of requests. This data statistic shows the success rate of HTTP requests (including read and write requests) initiated to COS by users.
  - Return code statistics
    Return code statistics are statistics of the number of HTTP request return codes that appear within the time range selected by the user, which can help users locate access situations.
  - Request type statistics
    Request type statistics categorize the number of requests by request type, mainly including 5 requests: GET, HEAD, PUT, POST, and DELETE.
- The upper part of the page displays the trend graph, and the lower part displays the statistics list. For each column with data in the statistics list, you can view the details of the success rate.
- The page supports exporting data to local storage. Click the export button on the right side of the date column to export data. There may be a delay in data, so refresh the page to reload it.

Data Monitoring
Viewing Data Overview

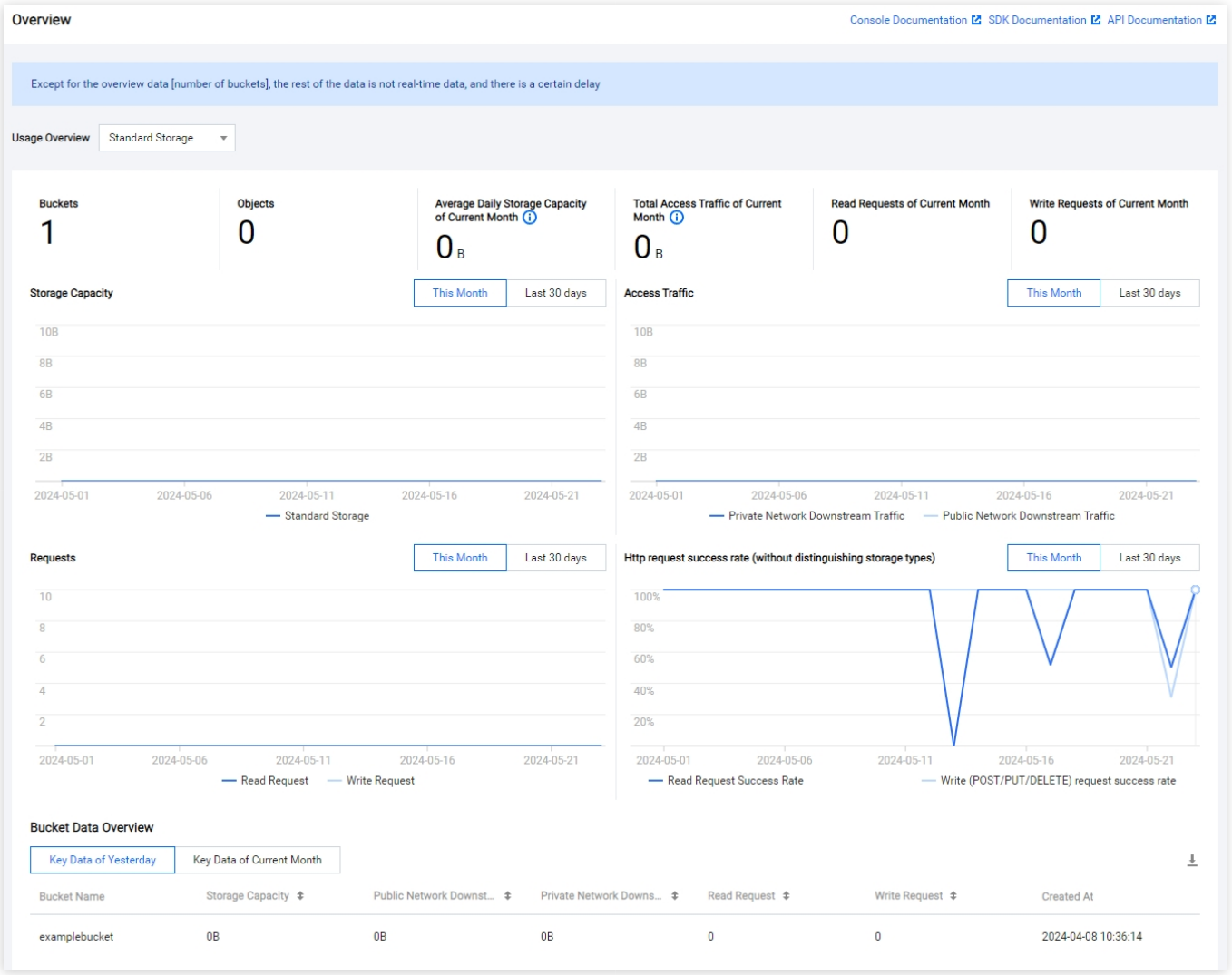
Overview

The COS console provides a storage data overview page, where you can view data such as the number of buckets, number of objects, storage usage, number of requests, and traffic.

> Note:
>
> The overview data is real-time data except for the number of buckets. The other data is not real-time data and is for reference only.

Viewing Data Folding Line Graphs

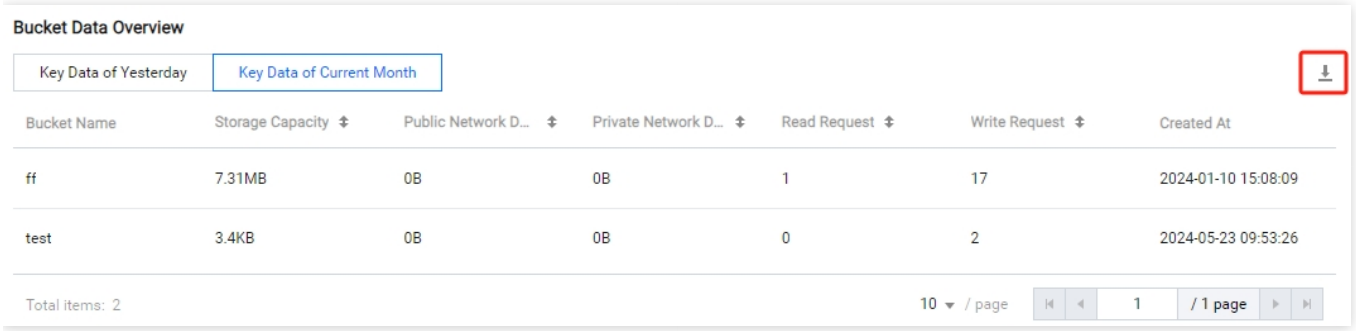
1. Log in to the COS console.
2. In the left navigation tree, select Overview.
3. On the Overview page, view basic usage statistics for the bucket.



- Number of buckets: The total number of buckets that have been created.
- Number of objects: You can view the total number of objects in STANDARD and STANDARD_IA classes by selecting the class in the drop-down list.
- Average daily storage usage for this month: You can view the daily storage usage of STANDARD and STANDARD_IA by selecting the class in the drop-down list.

Exporting Overview Data

On the Overview page, click the Download button on the right side of the bucket data overview to export all bucket data information from yesterday or this month, including storage capacity, public network downstream traffic, private network downstream traffic, and read and write requests.


Bucket Data Overview
Key Data of Yesterday
Key Data of Current Month
Bucket Name
Storage Capacity
Public Network D...
Private Network D...
Read Request
Write Request
Created At
ff
7.31MB
0B
0B
1
17
2024-01-10 15:08:09
test
3.4KB
0B
0B
0
2
2024-05-23 09:53:26
Total items: 2
10 / page
1
/ 1 page

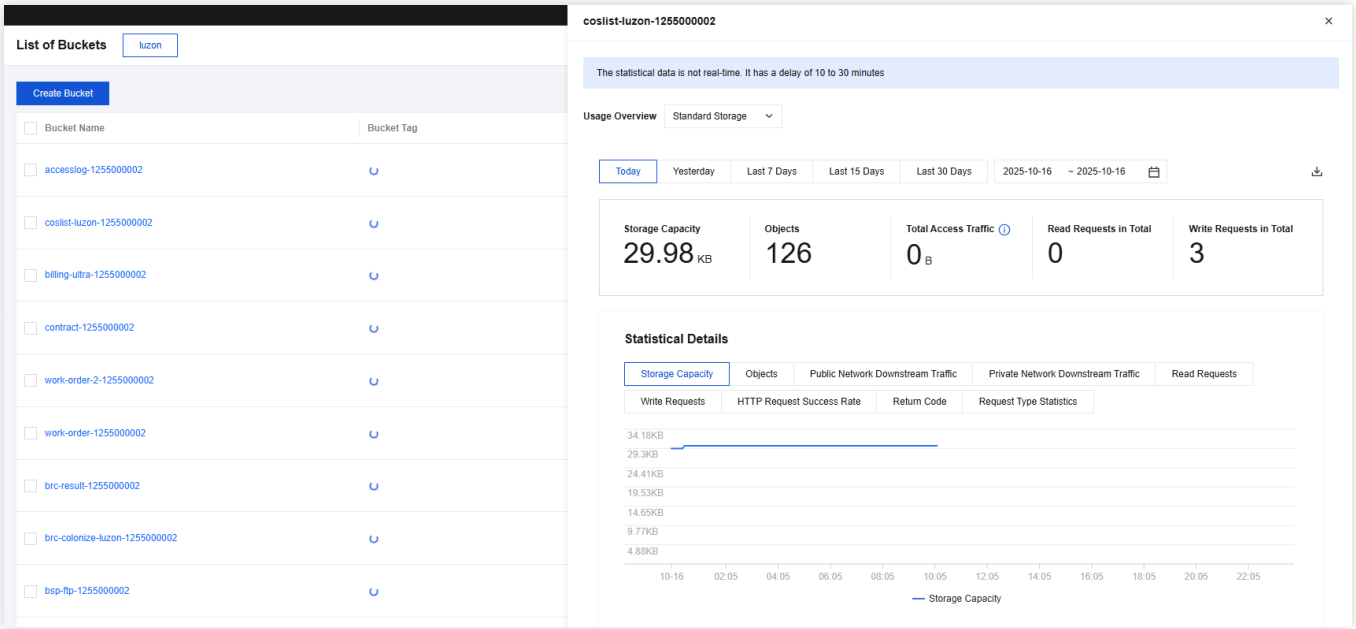
Viewing Data Monitoring

Overview

Cloud Object Storage (COS) provides the ability to monitor storage data. The COS data statistics monitoring window can display data such as request volume, traffic, and return code. You can use the monitoring data window to query the amount and trends of data for different storage types during different periods.

Directions

1. Log in to the COS console.
2. In the left navigation tree, select Bucket List.
3. On the bucket list page, find the bucket for which you want to collect data, click Monitoring on the right, and view the monitoring data of different storage types of the bucket.
   Support querying or downloading bucket usage information for standard storage and infrequent access storage in a specified period.



Setting Monitoring Alarms

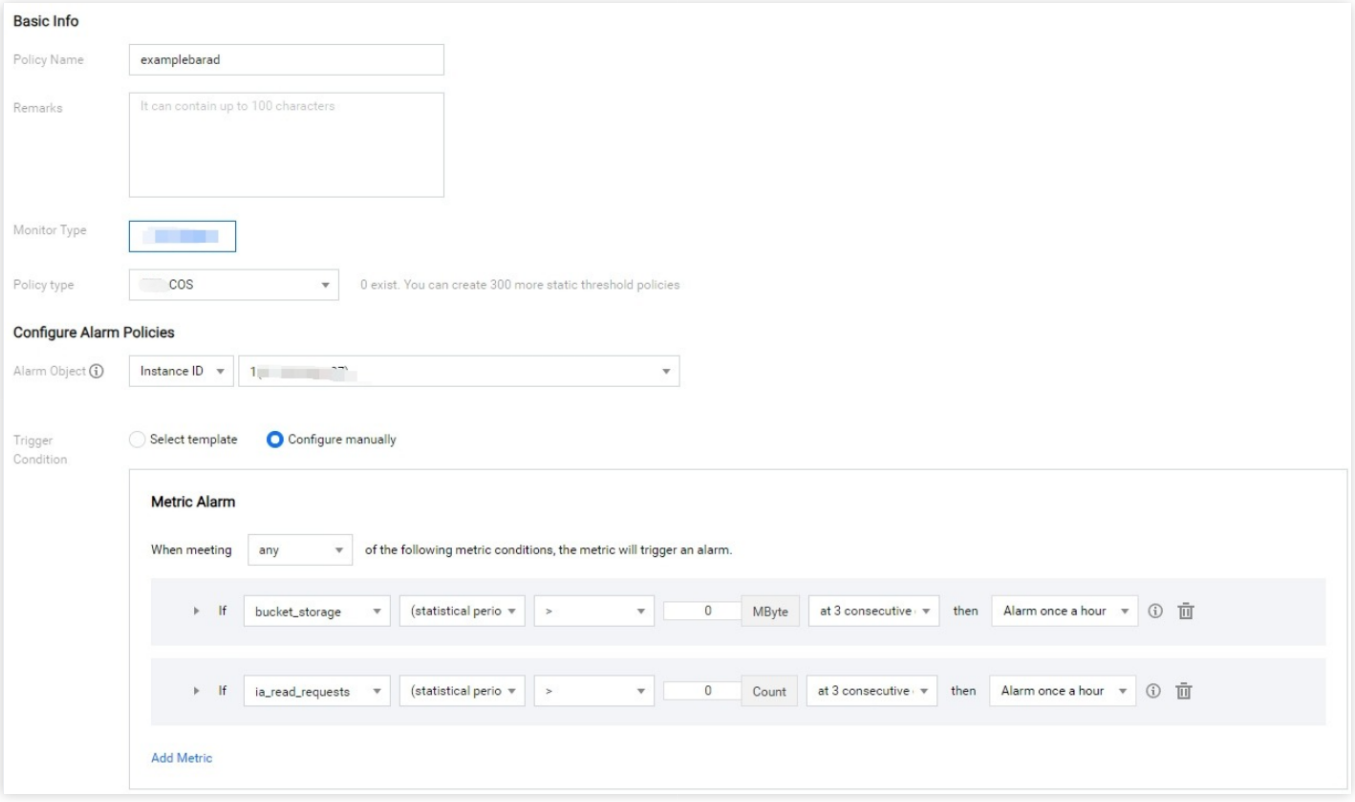
Overview

You can set threshold alarms for COS monitoring metrics through Cloud Monitor's alarm policies. Alarm policies include five essential components: alarm name, policy type, alarm trigger conditions, alarm objects, and alarm notification templates.

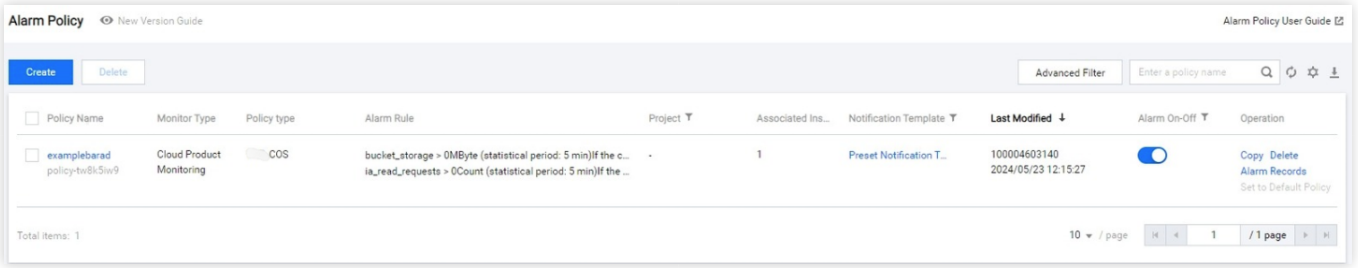
> Note:
>
> Cloud Monitoring is a service that monitors and alarms cloud service resources in real-time. Users can use Cloud Monitor to set alarm rules for COS monitoring metrics, view alarm records, and receive alarm notifications.

Directions

1. Log in to the Cloud Monitor console.
2. In the left navigation tree, select Alarm Configuration > Alarm Policy.
3. Click Create to start configuring the alarm policy.



4. After configuring the above information, click Complete, which means the COS alarm policy is successfully created.



5. Trigger according to the configured alarm policy. You can view the alarm records by entering the policy name.
6. When you need to stop monitoring alarms, just change the status of the corresponding rule to Off or click Delete to directly delete the alarm policy.

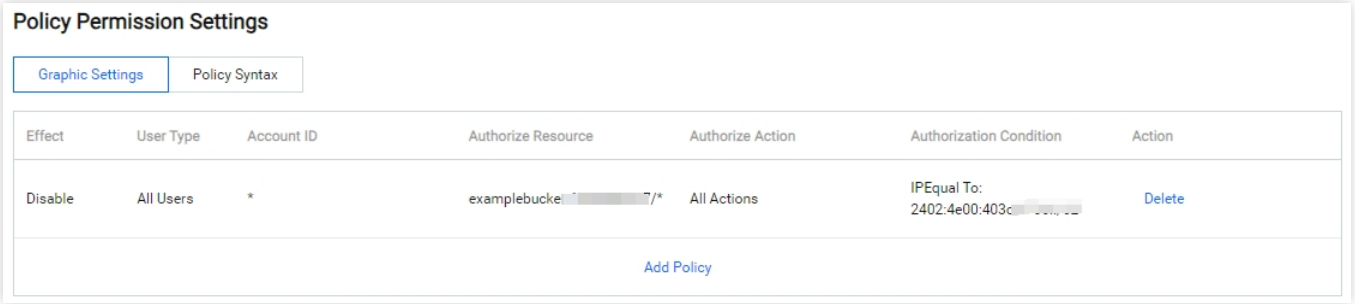
Advanced Features
Supporting IPV6 Access

Overview

The client supports private/public network IPV6 access and accesses COS via IPV6.

Setting IPV6 for Bucket Policy

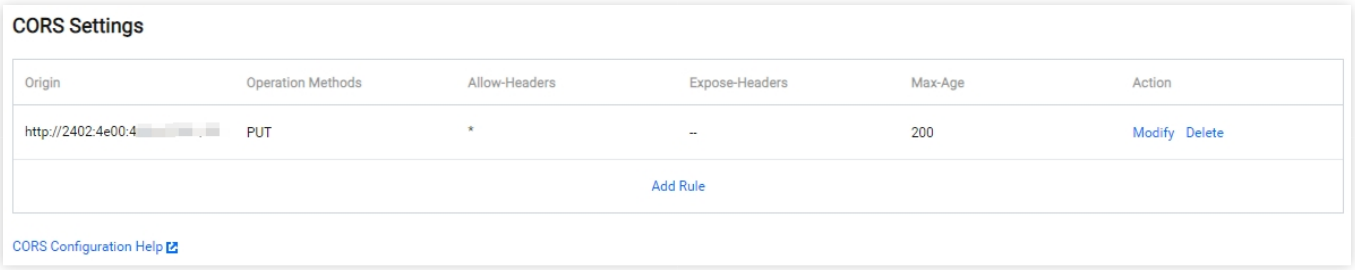
1. Go to the bucket list page, select Permission Management > Policy Permission Settings on the left, and click Add Policy.
2. Fill in the information to be set and set the condition to IP equal to the IPV6 condition value, as shown in the following figure:



At this time, all users will be prohibited from accessing all operations on the bucket through the IPv6 address.

Setting IPV6 for CORS

1. Go to the bucket file list page, select Security Management > CORS Settings, and click Add Rule.
2. Set the allowed source Origion to the IPV6 address, as shown in the following figure:

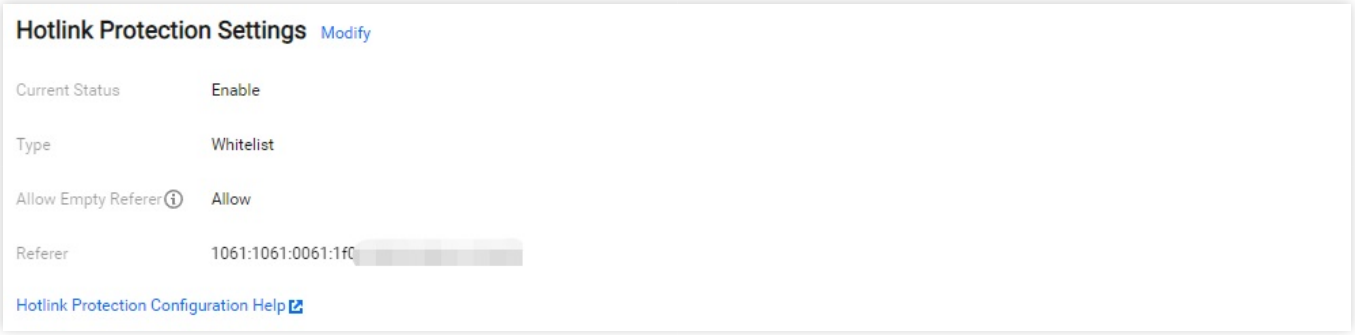


Cross-origin access will now be allowed for this domain.

Setting IPV6 for Hotlink Protection

1. Go to the bucket file list page, select Security Management > Hotlink Protection Settings, and click Modify.

2. Set Referer to the IPV6 address, as shown below:



The domain in the list will be allowed to access the bucket.

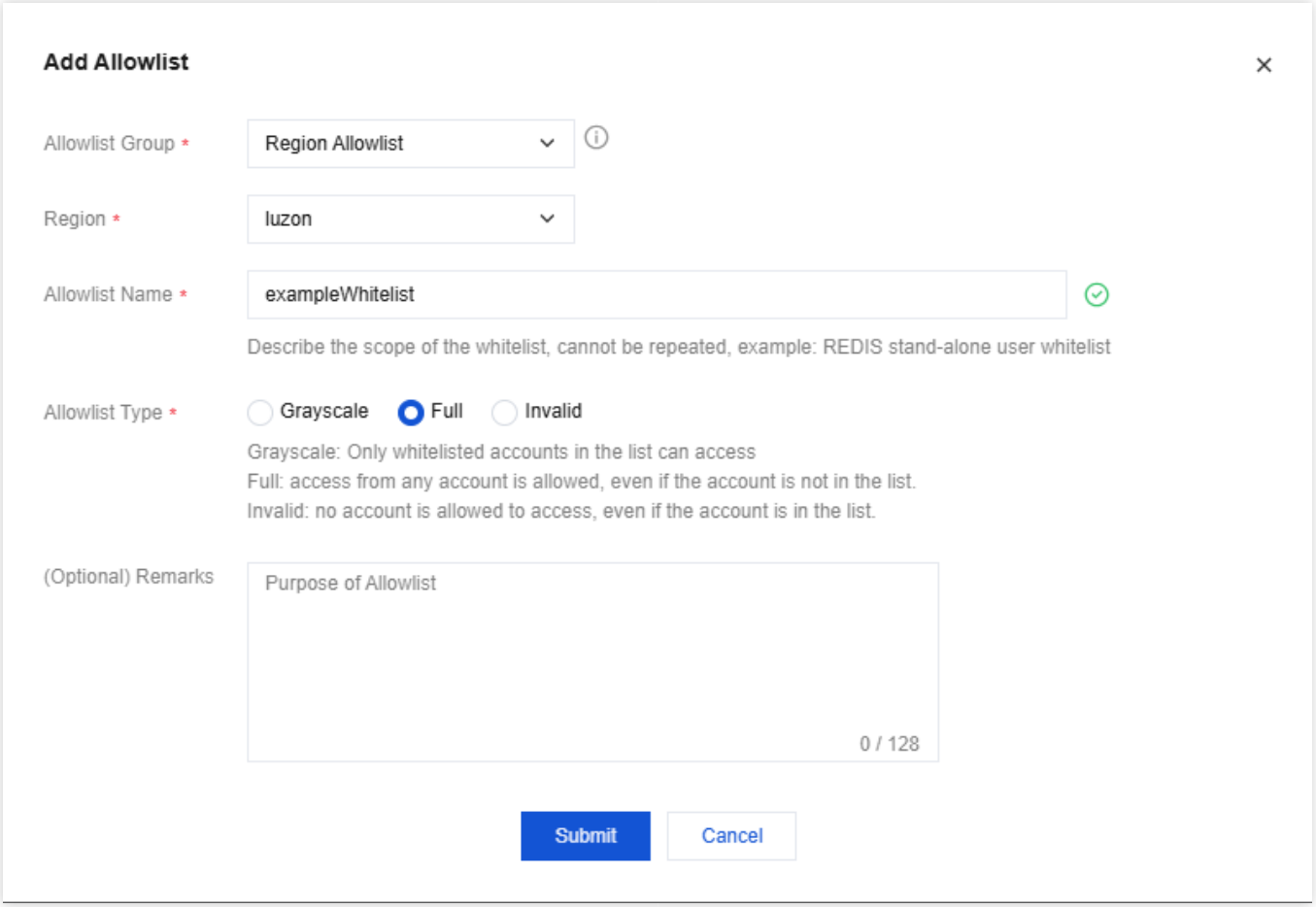
Region Allowlist

Overview

Support region allowlist. Only tenants (UIN) on the allowlist can access resources in this Region.

Directions

1. Log in to the operation side console.
2. Select Platform Operation > Customer Management.
3. In the left navigation tree, select Whitelist Management to add the corresponding allowlist type. For example, you can set a specified allowlist account as the effective allowlist type.



4. Only tenants (UIN) in the allowlist can access resources in this region. Otherwise, they have no access permissions.

Developer Guide
Bucket
Bucket Overview

Definition

A bucket is the carrier of objects, which can be understood as the container for storing objects. Users can manage the bucket and configure its attributes through various methods such as the cloud console, API, and SDK. For example, users can configure the bucket for static website hosting, configure bucket access permissions, and more.

Naming Specifications

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

- System generated numeric string APPID: This is automatically assigned by the system, and no user input is required. It is unique in Converge Cloud.
- A user-defined string of characters: A string of characters manually input by the user. The specifications are as follows.
  A custom string of characters in the following naming conventions:
  - Only lowercase English letters and numbers are allowed, namely a to z, 0 to 9, hyphens -, and their combinations.
  - The user-defined string of characters can be 1 to 50 characters.
  - A bucket name cannot start or end with -.

The following are examples of valid bucket names:

- mybucket123-1250000000
- 1-newproject-1250000000

Access Permission

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

Public Permissions

Public permissions include: Private Read/Write, Public Read/Private Write, and Public Read/Write. You can modify the access permissions in Permission Management of the bucket in the COS console.

- 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 control) for buckets by default. In COS, you can also add sub-accounts that are granted permissions to read/write data and permissions, and even full control, which is the highest permission level.

Related Description

- Object storage stores objects using a flat structure instead of folders.
- Each user account can create multiple buckets, with a number of up to 200 (in all regions), but there is no limit on the number of objects in a bucket.
- 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

Applicable Scenario

To start using Cloud Object Storage (COS), you need to create a bucket for using and managing objects. You can create a bucket through the console, APIs, or SDKs.

If the bucket does not exist, you can use the following code example to create a bucket in the specified region. The bucket supports the following parameters:

- Bucket: This is used to specify your full bucket name, such as testbuc-125235912 .
- Region: It is used to choose your service region of the cloud platform. Once you create it, you cannot move or modify the bucket.

Usage Instructions

Using the REST API

You can directly use the REST API to initiate a bucket creating request. For more information, see Put Bucket.

Using Java SDKs

COS supports AWS Java SDK. See Using AWS S3 SDK to Access COS to access COS, and operate according to the Put Bucket section in the official documentation of AWS Java SDK 2.20.0.

Deleting a Bucket

Applicable Scenario

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

> Note:
>
> 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.

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.

Usage Instructions

Using the REST API

Use the REST API to initiate a bucket deleting request. For more information, see Delete Bucket in the API documentation.

Using Java SDKs

COS supports AWS Java SDK. See Using AWS S3 SDK to Access COS to access COS, and operate according to the Delete Bucket section in the official documentation of AWS Java SDK 2.20.0.

Object
Object Overview

Definition

An object is the basic unit in Cloud Object Storage (COS) and is stored in a bucket (just like a photo stored in an album). You can manage objects through various means such as the Converge Cloud console, APIs, and SDKs. In API and SDK samples, an object name is in the format of <ObjectKey> .

> Note:
>
> Objects can be uploaded via Simple Upload or Multipart Upload.
>
> - 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.

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.
- Value: The size of the uploaded object.
- Metadata: A set of key-value pairs, which can be set when uploading an object.

Users can configure objects through the console.

- Searching for Objects
- Viewing Object Information
- Setting Object Access Permission
- Setting Custom Definition Headers

Object Key

Definition

An object in COS should have a valid object key, which is the unique identifier of the object in a bucket. For example, in an object's access address examplebucket-1250000000.<Endpoint>/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 (a-

z, or A-Z), digits (0-9), symbols (- , ! , _ , . , *), and their combinations to ensure optimal compatibility with other applications.
- The encoded length can be up to 850 bytes.
- 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.
  For example, when accessing document.doc , the object key is: document.doc . The actual percent-encoded value converted according to URL Encode rules is %e6%96%87%e6%a1%a3.doc .

The following are examples of valid object key names:

- my-organization
- my.great_photos-2016/01/me.jpg
- videos/2016/birthday/video.wmv

Special Characters

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. Special handling is required for these characters. The characters that may require special handling are as follows:

,	:	;	=
&	$	@	+
?	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. The characters to be avoided are as follows:

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

Related Description

Default 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, the access address for exampleobject.txt is examplebucket-1250000000.<Endpoint>/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. 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 delimiter / 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 Methods	Deletion	Result
Console.	Folder project	All objects with the object key prefix project/ will be deleted.
Console.	Object project/doc/a.txt	project and doc folders will remain
APIs and SDKs	Object project/ or project/doc/	Objects project/doc/*.txt will remain. If you want to delete the objects in the folder, use code traversal to delete them.

Object Metadata

Definition

The HTTP header of an object is a string sent by the server before transmitting HTML data to the browser using the HTTP protocol. By modifying the HTTP header, you can change the response format of the page or convey configuration information, such as modifying the cache time. Modifying the HTTP header of an object does not modify the object itself.

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 uploading or modification time.

Name	Description
Date	Current date and time
Content-Length	The HTTP request content length (in bytes) defined in RFC 2616, commonly used for operations of the PUT type API.
Last-Modified	The creation date or last modified date of an object (whichever is later).
Content-MD5	Base64-encoded 128-bit content MD5 checksum as defined in RFC 1864. This header is used to verify whether the file content has changed.
Authorization	Signature information that carries authentication information and is used to verify the validity of a request. For publicly readable files, this header is not needed.
x-cos-version-id	Object version. If versioning is enabled for a bucket, the version ID of the object will be returned.
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).
Expect	HTTP request content length (bytes) defined in RFC 2616.
Connection	Indicates the network connection status between the client and the server. Enumerated values: keep-alive, close.

Custom Metadata

This refers to the object's custom parameters, such as Content-Type, Cache-Control, Expires, and x-cos-meta-.

Name	Description
Cache-Control	Cache policy defined in RFC 2616. It is saved as object metadata.
Content-Disposition/ Encoding/Type	The content type (MIME) defined in RFC 2616 is saved as object metadata.
Expires	Expiration Time

Name	Description
x-cos-acl	Define Object's ACL attribute. Valid values: private, public-read-write, and public-read. Default value: private.
x-cos-grant-*	Grants the reading permission to the grantee.
x-cos-meta- *	// Allow user-defined header information, which will be returned as Object metadata, with a size limit of 2K.
x-cos-storage-class	Sets the storage class of the Object, with enumeration values: STANDARD, STANDARD_IA, ARCHIVE. Default value: STANDARD
x-cos-server-side-encryption	Specifies whether to enable server-side encryption for the object. If you use the COS primary key for encryption, fill in: AES256.

Object Subresource

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

An access control list contains an authorization list that can identify the authorized person and the permissions granted to it to achieve access control over objects. When an object is created, ACL will identify the object owner who has full control over the object. The user can retrieve the object ACL or replace it with an updated authorization list.

> Note:
>
> Any updates to the ACL require replacing the existing ACL.

Access Permission

COS supports setting two types of permissions for objects: Public Permission and User Permission.
Public Permission: Include private read/write, public read/private write, and public read/write.

- Private read/write: Only the creator of the bucket and authorized accounts have read/write permission on the objects in the bucket. No one else has read/write permissions. The default access permission of a bucket is private read/write, which is recommended.
- Public read and private write: Anyone (including anonymous visitors) has read permission to objects in the bucket, but only the bucket creator and authorized accounts have write permission to objects in the bucket.

- Public read and 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 control) for objects by default. In COS, you can also add sub-accounts that are granted permissions to read/write data and permissions, and even full control, which is the highest permission level.

Applicable Scenario

Set to allow public access to a specific object in a private read/write bucket, or set to require authentication to access a specific object in a public read/write bucket.

Uploading Objects
Simple Upload

Applicable Scenario

This operation is suitable for uploading an object smaller than 5 GB in a single request. For objects larger than 5 GB, you must use the multipart upload method.

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.

Usage Instructions

Using the REST API

Use REST APIs to initiate a simple upload request. For more information, see PUT Object.

Using Java SDKs

COS supports AWS Java SDK. See Using AWS S3 SDK to Access COS to access COS, and operate according to the PUT Object section in the official documentation of AWS Java SDK 2.20.0.

Multipart Upload

Applicable Scenario

Multipart upload is suitable for uploading large objects in poor or high-bandwidth network environments. The Cloud Object Storage (COS) console and SDKs help you split a single object into parts and upload them. You can also split the object yourself and call the API to upload each part separately. Multipart upload has the following advantages:

- In a poor network environment, a 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.
- Using multipart uploading, you can pause and resume uploading a single large object at any time. 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.
  When uploaded, these parts are numbered consecutively. You can upload each part separately or upload all the parts in any order. Eventually, COS will merge them into the object based on the order of the part numbers. If the transmission of any part fails, the failed part can be transmitted again without affecting other parts or the content as a whole. Generally, in a poor network environment, multipart upload is recommended for objects larger than 20 MB. In a high-bandwidth environment, multipart upload can be used to upload objects larger than 100 MB.

Usage Instructions

Using the REST API

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

- Initiate Multipart Upload
- Complete Multipart Upload
- Upload Part
- Abort Multipart Upload

Using Java SDKs

COS supports AWS Java SDK. See Using AWS S3 SDK to Access COS to access COS, and operate according to the PUT Object section in the official documentation of AWS Java SDK 2.20.0.

Uploading Pre-signed Authorized

Applicable Scenario

By default, buckets and objects are private. If you want a third party to be able to upload objects to your bucket but do not want them to use a CAM account or temporary token, you can use a pre-signed URL to submit the signature to the third party to complete the temporary upload operation. Anyone who receives a valid pre-signed URL can upload an object.

When pre-signing a URL, you can set the object key to be included in the signature to only allow uploads to the specified path. You can also specify the HTTP request method to restrict specific object operations, such as upload, download, and delete. You can also specify the validity period of the pre-signed URL in the program to ensure that the URL will not be used by unauthorized parties after the timeout.

Usage Instructions

Using the REST API

You can directly use the REST API to initiate a request to get an object. See the GET Object documentation for details.

Using Java SDKs

COS supports AWS Java SDK. See Using AWS S3 SDK to Access COS to access COS, and operate according to the Generating Pre-signed URLs section in the official documentation of AWS Java SDK 2.20.0.

Getting Objects
Simply Getting Objects

Applicable Scenario

You can directly initiate a request to get objects in Cloud Object Storage (COS). The following features are supported in getting objects:

- 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 part of the metadata values in the response using URL parameters, for example, the response value of Content-Disposition . The response headers that can be modified include the following:
- Content-Type
- Content-Language
- Expires
- Cache-Control
- Content-Disposition
- Content-Encoding

Usage Instructions

Using the REST API

You can directly use the REST API to initiate a request to get an object. See the GET Object documentation for details.

Using Java SDKs

COS supports AWS Java SDK. See Using AWS S3 SDK to Access COS to access COS, and operate according to the GET Object section in the official documentation of AWS Java SDK 2.20.0.

Downloading via Pre-Signed URL

Applicable Scenario

By default, buckets and objects are private. 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. You can also specify the validity period of the pre-signed URL in the program to ensure that the URL will not be used by unauthorized parties after the timeout.

Usage Instructions

Using Java SDKs

COS supports AWS Java SDK. See Using AWS S3 SDK to Access COS to access COS, and operate according to the Generating Pre-signed URLs section in the official documentation of AWS Java SDK 2.20.0.

Listing Object Keys

Applicable Scenario

The cloud platform's Cloud Object Storage (COS) supports listing object keys by a prefix. You can also use the / character to implement a hierarchical structure similar to the traditional file system. In COS, you can also use a separator 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 tapd object, but skip objects prefixed with a or other characters.

Add the / delimiter and objects keys will be reorganized based on this delimiter. You can combine prefixes and delimiters to achieve folder-like searching feature. For example, add the prefix parameter t and the delimiter /, object keys similar to tapd/file will be directly listed.

The cloud platform's COS supports storing an unlimited number of objects in a single bucket. Therefore, the object key list can be very large. For easy management, a maximum of 1,000 keys can be returned by a single list objects API, and a marker will be returned to inform you whether the list is truncated. You can send a series of listing object keys requests based on markers and separators to list all key values or find the content you need.

Usage Instructions

Using the REST API

Use REST APIs to initiate a request to get the object. For more information, see Get Bucket.

Using Java SDKs

COS supports AWS Java SDK. See Using AWS S3 SDK to Access COS to access COS, and operate according to the Get Bucket (List Objects) section in the official documentation of AWS Java SDK 2.20.0.

Replicating Objects Simple Copy

Applicable Scenario

In Cloud Object Storage (COS), you can create a replica of an object with a simple replication operation. You can replicate an object of up to 5 GB in a single operation. If an object is larger than 5 GB, you should use the multipart upload 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.
- 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.

Usage Instructions

Using the REST API

Use REST APIs to initiate an object copy request. For more information, see Put Object Copy documentation.

Using Java SDKs

COS supports AWS Java SDK. See Using AWS S3 SDK to Access COS to access COS, and operate according to the Put Object Copy section in the official documentation of AWS Java SDK 2.20.0.

Multipart Copy

Applicable Scenario

When you need to replicate an object that is larger than 5 GB, you need to use multipart replication. Use the multipart upload API to create an object and use the Part Copy feature to carry the x-cos-copy-source header to specify the source object that you want to replicate. The process includes the following steps:

1. Initialize an object for multipart upload.
2. Replicate the data of the source object; specify the x-cos-copy-range header. You can only replicate up to 5 GB at a time.
3. Complete the multipart upload.

You can use the SDKs provided by the cloud platform's Cloud Object Storage (COS) to easily complete a multipart replication operation.

Usage Instructions

Using Java SDKs

COS supports AWS Java SDK. See Using AWS S3 SDK to Access COS to access COS, and operate according to the File Copying section in the official documentation of AWS Java SDK 2.20.0.

Deleting an Object
Deleting an Object

Applicable Scenario

The cloud platform's Cloud Object Storage (COS) supports deleting one or multiple objects directly. If you only need to delete one object, just provide its name (object key) and call an API request to delete it.

Usage Instructions

Using the REST API

You can directly initiate a request to delete an object using the REST API. For more information, see Delete Object.

Using Java SDKs

COS supports AWS Java SDK. See Using AWS S3 SDK to Access COS to access COS and operate according to the Delete Object section in the official documentation of AWS Java SDK 2.20.0.

Data Management
Lifecycle Management
Lifecycle Overview

COS supports object-based lifecycle configuration. By sending specified description languages, it can make objects that meet the rules automatically execute some operations under the specified conditions.

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

Applicable Scenario

Logging

If you use COS to store log data, you can configure the lifecycle to automatically delete the log data in 2 years.

Support Notes

Supported Operations

- Deletion upon expiration: Set the expiration time of the object so that the object will be automatically deleted after expiration.

Supported Resources

- Specify by prefix: Objects that match the prefix rule will be processed according to the rule.
- Specify by version: non-current objects will be processed according to the rule.
- Specify by delete marker: The lifecycle feature can remove the marker automatically after all historical versions are deleted.
- Specify by incomplete multipart uploads: Process incomplete multipart uploads.

Supported Time Conditions

- Based on number of days: Specify after how many days since an object was last modified that an action will occur.
- Based on date: Specify the date when the action corresponding to the rule will occur.

Notes

Deletion upon Expiration

How It Works

When an object matches the specified lifecycle rule of deletion upon expiration, Converge 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.

Final 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.

> 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

For any configurations sent at any time, COS will start executing the operations at 0:00 the next day in local time. Since the operations are executed in an asynchronous queue, for objects uploaded after the rules are set, COS will usually complete the operations by 24:00 the next day.

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.

No billing commitment is provided for the effectiveness of the lifecycle execution, that is, the billing of the object will change when the lifecycle execution is completed.

No Size Limits

COS does not check the size of files and will unconditionally execute the conversion of objects according to the specified rules.

Lifecycle Configuration Elements

Basic Structure

Lifecycle configuration uses the XML description method and can be set with one or more lifecycle rules. The basic structure is shown below:

```
<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>
```

Here, each rule contains the following content:

- ID (optional): You can customize it to describe the content of the rule.
- Status: You can choose to set the rule status to Enabled or Disabled.
- Filter: You can use it to specify the filtering conditions of the objects to be operated.
- Operation: You can specify the operations that need to be performed on objects that match the above description.
- Time: You can specify the number of days (Days) based on the last modification date, or specify the objects that are modified before a specific date (Date).

Rule description

Filter Element

Based on All Objects in the Bucket

Specify an empty filter. This will apply to all objects in the bucket.

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

Based on a Specified 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 a Specified Object Tag

Specify some keys and values that match the object tags as the filter to execute the operation only on objects with the specific tag. For example, you can set it to filter out objects by the tag's 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>
```

Based on Multiple Filters

In Cloud Object Storage (COS), you can use the AND logic to combine multiple filters. For example, you can set it to filter out objects by the prefix logs/ as well as by the object tag's 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>
```

Operation Element

In a lifecycle rule, you can execute one or more operations on a group of objects that meet the conditions.

Transition Operation

Specify the Transition operation to transition an object from one storage type to another. If versioning is enabled for the bucket, the operation will only be executed on the current version. The shortest transition date can be set to 0 days. For example, you can set to transition an object to ARCHIVE storage type in 30 days:

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

Deletion upon Expiration

Specify the Expiration operation to delete an expired object that meets the rule. If visioning has never been enabled for the bucket, the operation will permanently delete the object. If visioning is enabled for the bucket, the operation will add a DeleteMarker marker to the expired object and set it as the current version. For example, you can set to delete an object in 30 days:

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

Incomplete Multipart Upload

Specify the AbortIncompleteMultipartUpload operation to maintain multipart upload tasks with the specified UploadId for a period of time before deleting them. Then, these tasks cannot be resumed or searched. For example, you can set to clear an incomplete multipart upload task in 7 days:

```
<AbortIncompleteMultipartUpload>
   <Days>7</Days>
</AbortIncompleteMultipartUpload>
```

Non-Current Version Object

In a versioning-enabled bucket, transition can only be executed on the latest version, and the expiration operation can only add deletion markers. Therefore, COS provides the following operations for objects of non-current versions:

Specify NoncurrentVersionTransition to transition non-current version objects to another storage type after a specified time. For example, you can set to transition the historical versions to ARCHIVE storage type in 30 days:

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

Specify NoncurrentVersionExpiration to delete non-current version objects after expiration in a specified time. For example, you can set to delete the historical versions in 30 days:

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

Specify ExpiredObjectDeleteMarker to clear the remaining delete marker when all historical versions of an object key are deleted and the latest version of the object is a DeleteMarker. For example, you can set it to remove the delete marker of an expired object in 31 days:

```
<ExpiredObjectDeleteMarker>
   <Days>31</Days>
</ExpiredObjectDeleteMarker>
```

Time Element

Based on the Number of Days

Use Days to specify the number of days. The time is calculated based on the last modification time of the object.

- For example, when you set to transition an object to ARCHIVE storage type in 0 days, and the object is uploaded at 23:55:00 GMT+8 on January 1, 2018. Then it will be added to the processing queue for storage type transition at 00:00:00 GMT+8 on January 2, 2018 and transitioned by 23:59:59 GMT+8 on January 2, 2018.
- For example, when you set to delete an object after expiration in 1 day, and the object is uploaded at 23:55:00 GMT+8 on January 1, 2018. Then it will be added to the processing queue for deletion after expiration at 00:00:00 GMT+8 on January 3, 2018 and deleted by 23:59:59 GMT+8 on January 3, 2018.

Based on a Specific Date

Use Date to specify the date. The operation will be executed on all objects that meet the filtering condition on the specified date. Currently, only GMT+8 time zone is supported. The time is set to 00:00 GMT+8 in the ISO8601 format.
For example, January 1, 2018, is described as: 2018-01-01T00:00:00+08:00.

Configuring Lifecycle

Applicable Scenario

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

- Storage type transition: Transition objects to the STANDARD_IA or ARCHIVE storage type after a specified period of time.
- Deletion upon expiration: Set the expiration time of the object so that the object will be automatically deleted after expiration.

Usage Instructions

Using the COS Console

Configure the lifecycle via the COS console. For more information, see Setting the Lifecycle in Console Guide.

Using the REST API

You can use the REST API directly to configure and manage the lifecycle of objects in a bucket. For more information, see the following API documentation:

- PUT Bucket lifecycle
- GET Buket lifecycle
- DELETE Bucket lifecycle

Static Website Hosting

Basic Concepts

A static website refers to a website that contains static content (such as HTML) or client scripts. Users can configure the static website through the console for the storage bucket bound with custom domains. The content of dynamic websites contains server-side scripts such as PHP, JSP, or ASP.NET, which need to rely on server-side processing. COS supports hosting static websites but does not support server-side scripting. When you need to deploy dynamic websites, it is recommended to use CVM for server-side code deployment.

Example

The 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.city.yfm4.fsphere.cn/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.city.yfm4.fsphere.cn/index.html

Index Document

Index document, which is the homepage of a static website, is the webpage returned when a user requests the root directory or any subdirectory. This page is typically named index.html .
When a user accesses a static website using the bucket access domain (e.g., https://examplebucketbucket-1250000000.cos-website.city.yfm4.fsphere.cn) without specifying a particular page, the Web server will return the homepage in this case.

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 document 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.city.yfm4.fsphere.cn/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.

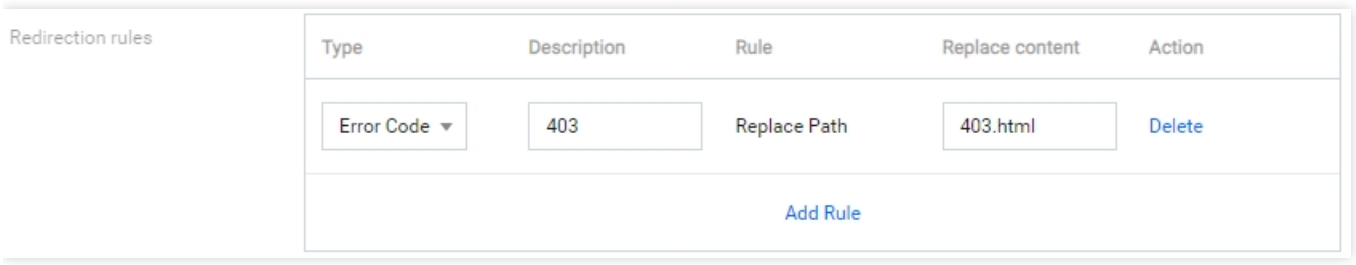
https://examplebucket-1250000000.cos-website.city.yfm4.fsphere.cn/webpage.html

Redirection Rules

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

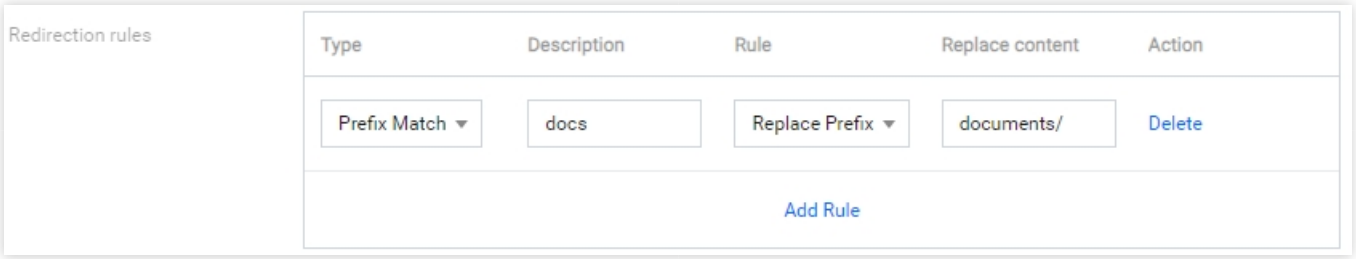
Configuring Error Code Redirection

If the webpage.html document is set to Private Read/Write public assess permission, when a user tries to access it, a 403. 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.

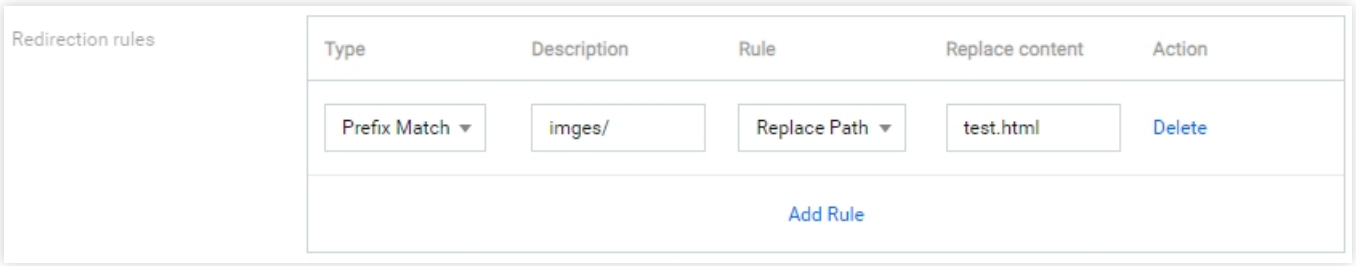


Configuring Prefix Match

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/.



2. you can add a redirection rule to redirect requests for any object with the prefix images/ to test.html . You can add a redirection rule to redirect requests for any object with the prefix images/ to test.html .



Inventory Overview

Inventory Introduction

Inventory is a feature that helps you manage objects in a bucket, and can be planned to replace the operations of the COS synchronous List API. Based on your inventory task configuration, COS can scan daily or weekly a specified object or objects with the same prefix in your bucket, export an inventory report, and save it as a CSV file to a bucket you specify. The file will list the stored objects and their metadata, and record the object attributes you need based on your configuration.
You can use the inventory feature for various purposes, including but not limited to the following basic ones:

- 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 tasks in one bucket. During the execution of these tasks, the object content will not be read directly, and only the attribute information such as object metadata will be scanned.

Inventory Parameters

After you configure an inventory task, COS will regularly scan the specified objects in your bucket based on your configuration, and export an inventory report, which is a CSV file. Currently, the following information can be recorded in an inventory report:

Inventory Information	Description
AppID	Account ID.
Bucket	Name of the bucket where the inventory task is executed.
fileFormat	File format.
listObjectCount	Number of listed objects. This item will be used for billing.
listStorageSize	Size of listed objects.
filterObjectCount	Number of filtered objects.
filterStorageSize	Size of filtered objects.

Inventory Information	Description
Key	Name of an object file in the bucket. When the CSV format file is used, the object file name is URL-encoded and can only be used after it is decoded.
VersionId	Version ID of an object. After versioning is enabled for the bucket, COS will specify a version number for the object added to the bucket. This field will not be included if the list is only for the current version of the object.
IsLatest	Set to True if the object is the latest version. If the list is only for the current version of the object. This field will not be included if the list is only for the current version of the object.
IsDeleteMarker	Set to True if the object is a delete marker. This field will not be included if the list is only for the current version of the object.
Size	Object size (in bytes).
LastModifiedDate	The last modification date of the object (whichever is later).
ETag	Entity tag. It is the hash value of the object. The ETag only reflects changes to the object content, but not changes to the object metadata. The ETag may or may not be the MD5 checksum of the object data, and this depends on how the object is created and encrypted.
StorageClass	Storage class of the object.
IsMultipartUploaded	Set to True if the object is uploaded through multipart upload. <!--For more information, see Multipart Upload.
Replicationstatus	Marks the status of the source and replication files during the object replication. Markers for the source file: PENDING (waiting to be replicated), COMPLETED (replication completed), FAILED (replication failed); Markers for the replication file: REPLICA (replication completed, with a replication file generated). For more information, see Replication Behavior Description.
Tag	Object tag.

Inventory Configuration

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

- Source bucket: The bucket for which you want to enable the inventory feature.
    - Contains objects listed in the inventory.

    - Contains the configuration of the inventory.
- Destination bucket: The bucket where you want to store the inventory.
    - Contains an inventory list file.
    - Contains the Manifest files that describe the locations of the inventory list file.

      You need to follow the steps below to configure an inventory:

Specifying the Object Information to Be Analyzed in the Source Bucket

You need to tell COS the object information to be analyzed. Therefore, when configuring the inventory feature, you need to configure the following information in the source bucket:

- Select object versions: List all the versions of objects or list only the current version. If you select to list all the versions of an object, COS will record all historical versions of the object with the same name in the inventory report. If you select to list only the current version, COS will record only the latest version of the object.
- Configure the attributes of the objects to be analyzed: You need to tell COS what information from the object attributes needs to be recorded in the inventory report. Currently, supported object attributes include account ID, source bucket name, object file name, object version ID, whether it is of the latest version, whether it is a delete marker, object size, object's last modification date, ETag, object's storage class, cross-region replication tag, and whether it is a part in multipart upload.

Configuring the Storage Information for the Inventory Report

You need to tell COS at what frequency to export the inventory report, to which bucket you want the report to be stored, and whether the report needs to be encrypted. You need to configure the following information:

- Select an inventory export frequency: Daily or weekly. You can use this configuration to tell COS at what frequency it should execute the inventory feature.
- Select an inventory encryption mode: No encryption or SSE-COS. If you select SSE-COS encryption, we will encrypt the generated inventory report.
- Configure the inventory output location: You need to specify the bucket where to store the inventory report.

  > Note
  >
  > The destination and source buckets should be in the same region, and can be the same bucket.

Usage Instructions

Configuring Inventory via the Console

To learn how to configure the inventory feature via the console, see Activate Inventory Feature in the

console documentation.

Configuring Inventory via APIs

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

1. Log in to Tenant Side as the primary account, navigate to the Inventory Settings page of any bucket, and perform service authorization. Authorization is only required once for the same account.
2. Enable the inventory feature.

1. Enabling the Inventory Feature

Call the API to enable the inventory feature. For more information about the API, see PUT Bucket inventory. Here, the destination bucket to store the inventory file should be in the same region as the source bucket.

Inventory Report Storage Path

The inventory report and the relevant manifest files will be published to the destination bucket, in which the inventory report will be published in the following path:

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

The manifest files will be published 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: The "destination prefix" you set when you configure the inventory. It can be used to group all inventory reports in a public location in the destination bucket.
- source-bucket: 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 own inventory reports to the same destination bucket.
- config-ID: The "inventory name" you set when you configure the inventory. When multiple inventory reports are set in the same source bucket and sent to the same destination bucket, config-ID can be used to distinguish among different inventory reports.
- YYYYMMDD: Timestamp, including the start time and date of the bucket scan when the inventory report is generated.
- manifest.json: Means the manifest file.
- manifest.checksum: Means the MD5 of the content of the manifest.json file.
  Here, the manifest files include two files: manifest.json and manifest.checksum.

  > Description

> The manifest files are described as follows:
>
> - Both manifest.json and manifest.checksum are manifest files. manifest.json describes the location of the inventory report, and manifest.checksum serves as the MD5 of the content of the manifest.json file. Each time a new inventory report is delivered, it comes with a new set of manifest files.
> - Each manifest contained in manifest.json provides the basic information of the inventory, such as the metadata. Such information includes the following items:
>   - Name of the source bucket.
>   - Name of the destination bucket.
>   - Version of the inventory.
>   - Timestamp, including the start time and date of the bucket scan 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.

A sample of the manifest in the manifest.json file for a CSV-format inventory is shown below:


```
{
 "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, IsMultipartUploaded, Re
plicationStatus",
 "files": [
  {
   "key": "cos_bucket_inventory/1250000000/examplebucket/inventory01/04d73d9debc73d9f0bf85af4
61abde6c.csv.gz",
   "size": "502",
   "md5Checksum": "7d40288a09c25b302ad6cb5fced54f35"
  }
 ]
}
```


Inventory Consistency

Inventory report by COS provides eventual consistency for PUT of both new objects and overwrites and for DELETE. Therefore, the latest added or deleted objects might not be included in the inventory report. For example, if you upload or delete an object when COS is executing an inventory task you configure, you might not see the results of your upload or deletion operations 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

Overview

A bucket tag is a key-value pair (key = value), where the tag key and tag value are connected by an equal sign "=", for example, group = IT. It can be used as an identifier to manage buckets, helping you 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.
- Tag keys can contain characters encoded in UTF-8, spaces, digits, and 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

- Tag values can contain characters encoded in UTF-8, spaces, digits, and 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 can have up to 1,000 different keys.
  - A key has at most 1,000 values.
  - Multiple same keys are not allowed in the same bucket.

Usage Instructions

You can set bucket tags via the console or APIs.

Using the COS Console

If you need to use the COS console to set bucket tags, see Setting Bucket Tags in the document Console

Guide.

Using the REST API

You can use the following APIs to manage bucket tags directly:

- PUT Bucket tagging
- GET Bucket tagging
- DELETE Bucket tagging

Object Tag Overview

Feature Overview

The object tagging feature is used to help you group and manage objects in your bucket by adding a key-value pair as an identifier for objects. An object tag consists of a tag key (tagKey), a tag value (tagValue), and an equal sign = , for example, group = IT . You can set, query, and delete tags for a specified object.

Usage Instructions

Using the COS Console

You can manage object tags via the COS console.

Using the REST API

You can manage object tags via the following APIs:

- PUT Object tagging
- GET Object tagging
- DELETE Object tagging

Specifications and Limits

Limits on Tag Keys

- Tag keys can contain characters encoded in UTF-8, spaces, digits [0–9], and special characters including + - = ._ : / @ .
- Tag keys can contain 1 to 127 characters (encoded in UTF-8).
- Tag keys are case-sensitive.

Limits on Tag Values

- Tag values can contain characters encoded in UTF-8, spaces, digits [0–9], and special characters including + - = ._ : / @ .
- Tag values can contain 1 to 255 characters (encoded in UTF-8).
- Tag values are case-sensitive.

Limits on the Number of Tags

- Object dimension: Up to 10 different object tags per object.
- Tag dimension: unlimited.

Log Management
Logging Overview

Overview

The log management feature allows you to record the detailed access information of a specified source bucket and save the information as log files to a specified bucket for better bucket management.

In a destination bucket, the log path is:

Destination bucket/path prefix{YYYY}/{MM}/{DD}/{time}_{random}_{index}.

A log is generated every 5 minutes, with one record per line. Each record contains multiple fields, which are separated by a space. Note that the size of a log file is up to 256 MB. If the log you generate within the 5 minutes is larger than 256 MB, it will be split into several log files. The currently supported logging fields are shown below:

No.	Field	Description	Example
1	eventVersion	Recorded version.	1.0
2	bucketName	Bucket Name	examplebucket-1250000000
3	qcsRegion	Request region.	ap-city
4	eventTime	Event time (request end time, which is a timestamp in UTC+0 time zone).	2018-12-01T11:02:33Z
5	eventSource	User access domain name.	examplebucket-1250000000.cos.ap-city.xxx.com
6	eventName	Event Name	UploadPart
7	remoteIp	Source IP address	192.168.0.1

No.	Field	Description	Example
8	userSecretKeyId	User access KeyId.	AKIDNYVCdoJQyGJ5brTf
9	reservedFiled	Reserved field.	Reserved field, which is displayed as - .
10	reqBytesSent	Request bytes (bytes).	83886080
11	deltaDataSize	Change in storage made by the request (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	Returned bytes.	197
19	resTotalTime	Total time taken for a request (in milliseconds. It is equal to the time between the last byte of the response and the first byte of the request).	4295
20	logSourceType	Source type of the log.	USER (user access request).

No.	Field	Description	Example
21	storageClass	Storage Type	STANDARD, STANDARD_IA, and ARCHIVE.
22	accountId	ID of the bucket owner.	100000000001
23	resTurnAroundTime	Time taken for the request server side (in milliseconds. It is equal to the time between the first byte of the response and the last byte of the request).	4295
24	requester	Requester.	Root account ID: sub-account ID. If the requester is anonymous, it is displayed as - .
25	requestId	Request ID.	NWQ1ZjY4MTBfMjZiMjU4NjRfOWI1N180NDBiYTY=
26	objectSize	Object size (bytes).	808. If you use multipart upload, the objectSize field will only be displayed when the upload is completed. During the multipart upload process, this field is displayed as - .
27	versionId	Object version ID.	Random string.
28	targetStorageClass	Destination storage type. This field is recorded for requests that initiate a replication operation.	STANDARD, STANDARD_IA, and ARCHIVE.
29	referer	HTTP referer of the request.	*.example.com or 111.111.111.1.
30	requestUri	Request URI	"GET /fdgfdgsf%20/%E6%B5%AE%E7%82%B9%E6%95%B0 HTTP/1.1"

No.	Field	Description	Example
31	vpcId	Whether the request is a VPC request.	0/1

> Note:
>
> The destination bucket that stores logs can be the source bucket itself, but this is not recommended.
>
> Based on user needs and business development, COS may add new fields to the access logs. Be sure to handle this accordingly when you parse the logs.

Enabling Log Management

You can quickly enable the log management feature in the tenant console.

Restrictions on Log Management

The log management feature allows you to record the detailed access information of a specified source bucket and save the information as log files to a specified bucket for better bucket management.

The current use limits on the log management feature are shown below:

- Limit on shipping frequency: A log is generated every 5 minutes.
- Limit on the size of log files to be shipped: The log file to be shipped each time can be up to 256 MB. If a file exceeds this limit, its surplus part will be shipped as a new file.
- Limit on the format of logs to be shipped: There is one record per line. Each record contains multiple fields, which are separated by a space.
- Limit on the fields to be shipped: For more information on the fields to be shipped, see the description in "Logging Overview".
- Limit on 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 you initiate to upload, download, and delete an object, as well as to create and delete a bucket, and to modify the bucket configuration.

Content Not Recorded in a Log

- Offline origin-pull requests: After you configure origin-pull, if there are no objects in COS, the data will be downloaded from the origin server you specify. This download operation is an offline origin-pull request and will not be recorded in the log.
- Operations of redirects within a static website: After you configure redirection in the static website feature, you might be redirected to another page when accessing index.html. These redirection operations will not be recorded in the log.
- Operations of lifecycle transitions and deletions: After you configure the lifecycle feature to transition or delete objects after expiration, these transition and deletion operations will be implemented by the COS backend and will not be recorded in the log.
- Operations of listing objects and uploading inventory reports: The inventory feature periodically lists all or specified objects in your buckets for you and ships the generated inventory report to your buckets. The operations of listing objects and shipping inventory reports will not be recorded in the log.
- Operations of cross-region object replication: Cross-region replication requires getting an object

from the source bucket and uploading the object to the destination bucket. These operations are implemented by the COS backend and will not be recorded in the log.

- Operations of object downloads in the COS Select feature: The COS Select feature helps you search for an object, but the search can only be performed after the data is obtained from a storage device. The operations of object downloads are implemented by the COS backend and will not be recorded in the log.

Data Disaster Recovery
Versioning
Versioning Overview

Overview

Versioning is used to store multiple versions of an object in the same bucket. For example, in one bucket, you can store multiple objects with the same object key "picture.jpg", but they have different version IDs such as 100000, 100101, and 120002. After enabling the versioning feature for a bucket, you can query, delete, or restore the objects stored in the bucket by their version IDs. This enables you to restore lost data caused by accidental deletion or application faults. For example, when you delete an object with visioning enabled:

- If you need to delete the object (not permanently), COS will insert a delete marker for the deleted object, and the marker will serve as the current object version. You can restore the object to its previous version based on Delete Marker.
- If you need to replace the object, COS will insert a new version ID for your recently uploaded object. In this case, you can also restore the original object based on the version ID.

Versioning Status

There are three versioning statuses for a bucket: Versioning not enabled, versioning enabled, and versioning suspended.

- Versioning not enabled: The initial versioning status of a bucket by default. In this status, the versioning feature is disabled.
- Versioning enabled: The versioning feature of a bucket is enabled. In this status, versioning is enabled. The versioning status will be applied to All Objects in the bucket. After you enable versioning for the bucket for the first time, the recently uploaded objects in the bucket will have 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:
  >
  > i. Once versioning is enabled for a bucket, it cannot be disabled (the initial status). However, versioning for the bucket can be suspended, so that the recently uploaded objects

> afterward will not generate multiple versions.
> ii. Before versioning is enabled, the version IDs of all the objects stored in the bucket are null.
> iii. Enabling or suspending versioning will change how COS makes requests when it handles the objects, but will not change the objects themselves.
> iv. Only the root account and authorized sub-accounts can suspend versioning for a bucket.

Managing Objects Under Versioning Status

Buckets are in different versioning statuses. You can upload, query, and delete objects in buckets in different statuses. In addition to the disabled, enabled, and suspended versioning statuses, operations for querying objects in a bucket and deleting objects also include operations without specifying a Version ID and operations with specifying a Version ID.

- When the versioning status is disabled, operations such as uploading, querying, and deleting objects remain unchanged.
- When the versioning status is enabled or suspended, operations such as uploading, querying, and deleting objects are different from previous methods in that the Version ID is introduced. The concept of Delete Marker is also included in the operation of deleting objects.

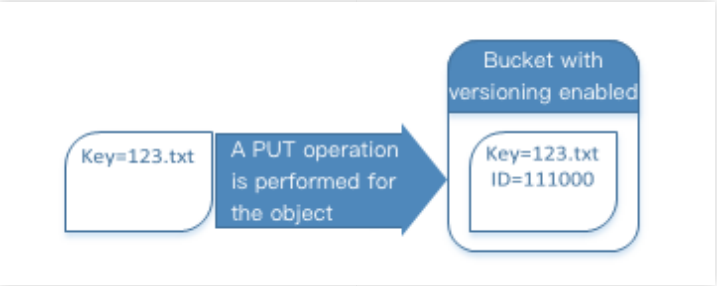
Managing Objects with Versioning Enabled

Before bucket versioning is enabled, the Version IDs of objects already stored in the bucket are null. After enabling versioning, existing objects in the bucket will not be changed. It only changes the way COS handles existing objects (such as the request method). At this point, newly added objects with the same name exist in the same bucket as different versions.

> Note:
>
> Users upload objects in the same way to both versioning-disabled and versioning-enabled buckets, but their Version IDs are different. In the second method, the COS assigns a specific Version ID to the object, while the Version ID of the object uploaded in the first method is always null.

Uploading an Object

After versioning is enabled for a bucket, when a user performs a PUT, POST, or COPY operation, COS automatically adds a unique Version ID to the object placed in the bucket.
As shown in the image below, when you upload an object in a bucket with versioning enabled, Object Storage adds a unique version ID to the object.



Listing Versioning Objects

COS stores the version information of objects in the versions parameter associated with the bucket. COS returns object versions in the order of storage time, returning the most recently stored version first.

Querying All Versions of a Specific Object

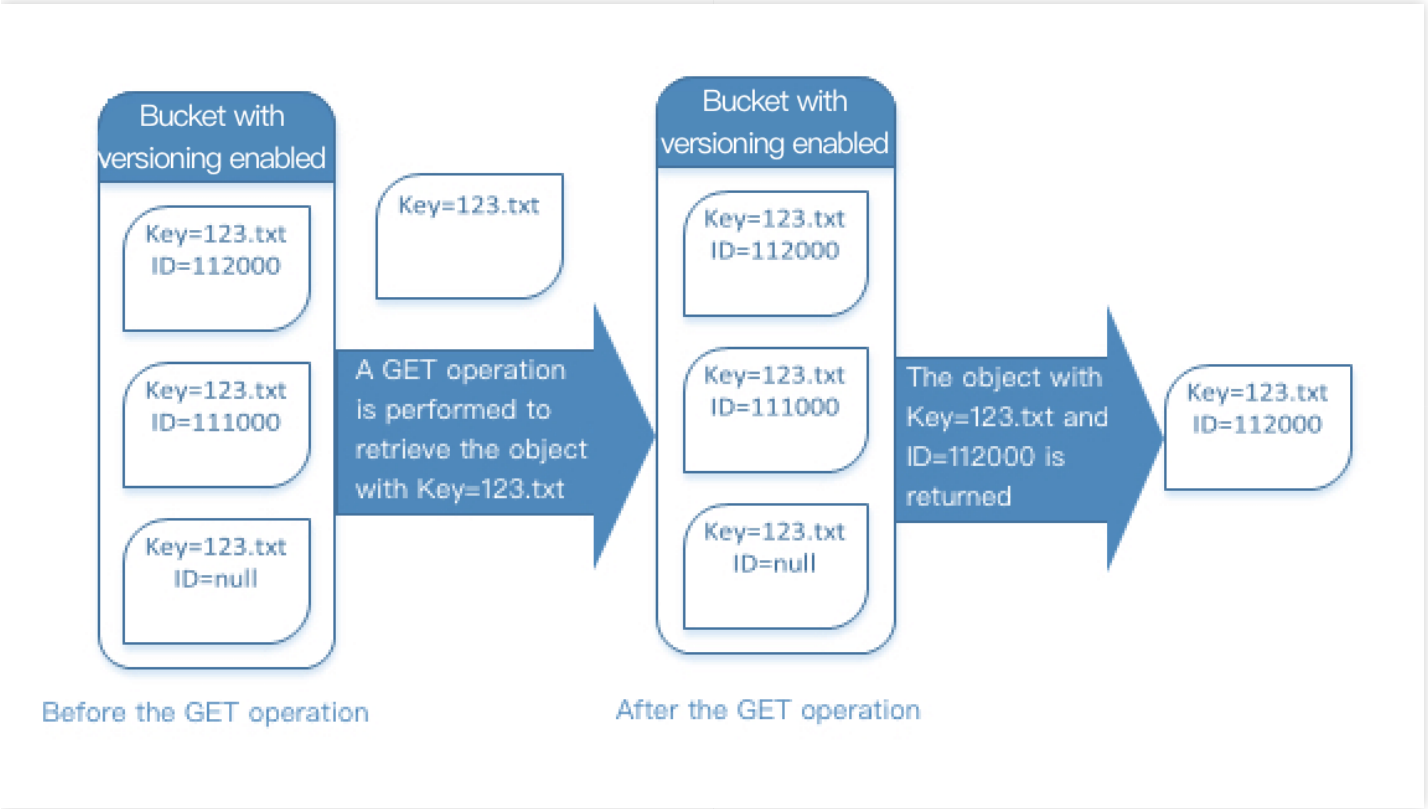
You can use the versions parameter and the prefix request parameter to query all versions of an object through the following procedures. For more prefix information, see the GET Bucket Object Versions documentation.

You can query all versions of a specific object. The request sample is shown below:

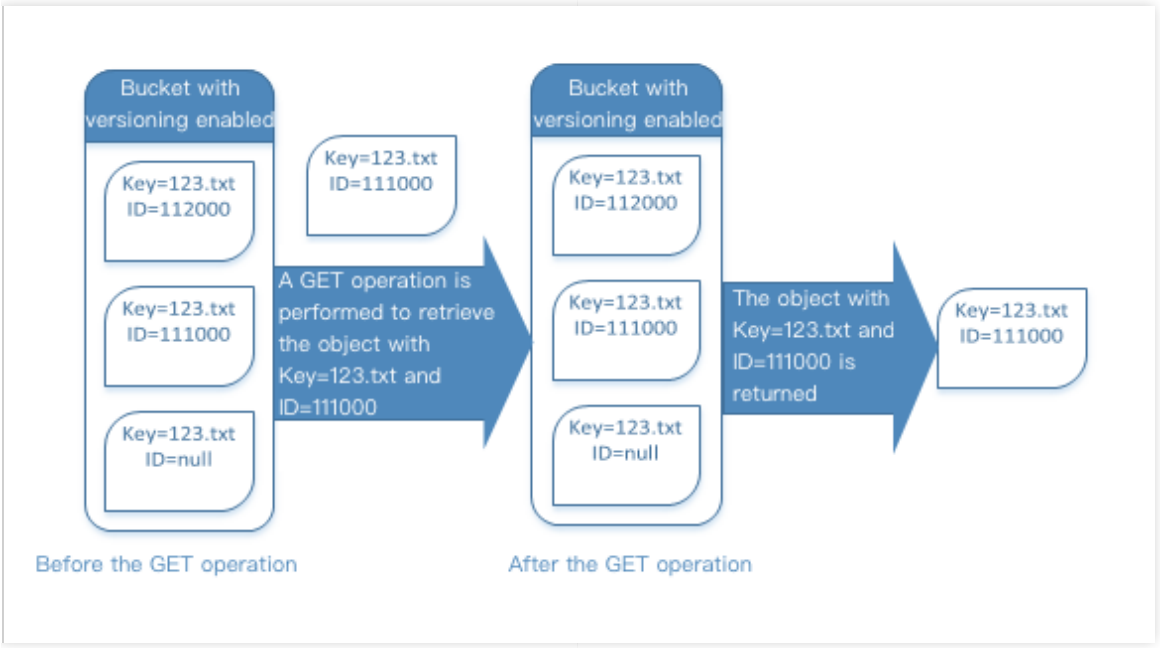
```
GET /?versions&prefix=OjectKey HTTP/1.1
```

Querying the Metadata Version

When a user uses a GET request without specifying a Version ID, the current version of the object will be queried. As shown in the following figure, the GET request will return the current version (latest version) of the 123.txt object.



If the user specifies the VersionId when using the GET request, the object with the specified Version ID is queried. As shown in the following figure, the GET VersionId request queries the object of the specified version (which can be the current version).



Querying the Metadata of Object Versions

If you need to query only the metadata of an object (not its contents), you can use the HEAD operation. By

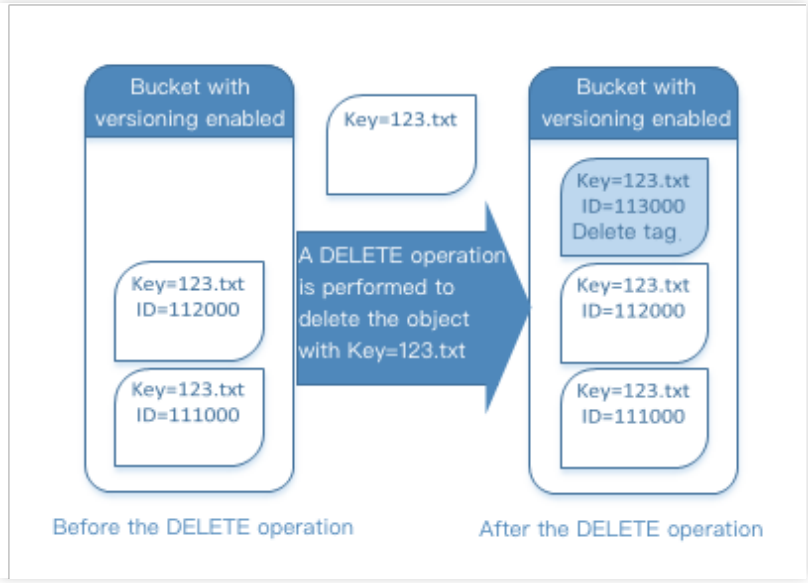
default, you will get the latest version of the metadata. If you want to query the metadata of a specific object version, you need to specify its Version ID when sending a request.
The steps to query the metadata of a specified version of an object are as follows:

- Set VersionId to the Version ID of the object metadata being queried.
- Send a HEAD operation request with the specified VersionId.

Deleting an Object

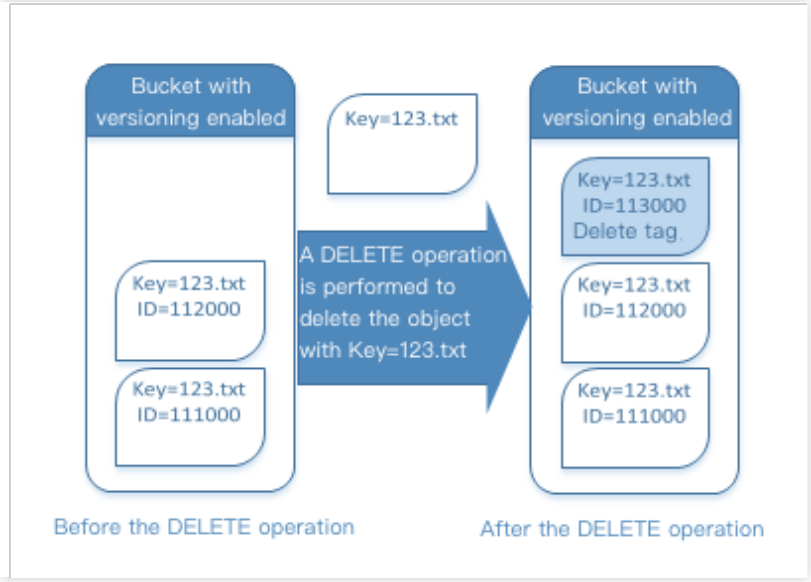
You can delete unnecessary object versions at any time as needed. When versioning is enabled, there are two scenes for users to use the DELETE request:

1. Deleting objects: Perform a simple DELETE operation without users specifying a version ID.
   This overview is similar to putting the deleted objects into the "recycle bin," but not completely removing them. If needed later, users can still retrieve the data.
   As shown below, if users do not specify a version ID during the DELETE operation, the object with Key=123.txt will not be deleted actually. Instead, a new deletion marker is inserted, and a new version ID is added.



> Note:
>
> COS will insert a delete marker with a new version ID for the deleted object in the storage bucket. This delete marker will become the current version of the deleted object. When you attempt to perform a GET operation on an object with a delete marker, COS will assume that the objects do not exist and will return a 404 error.

2. The user specifies the Version ID and executes an operation to delete the object version. In this scene, the versioning object can be permanently deleted.



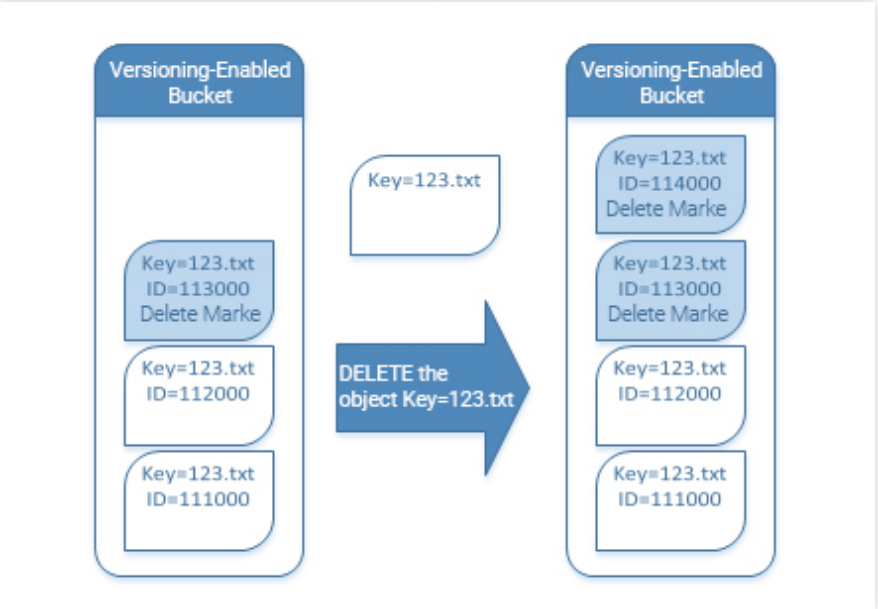
Deleting Markers

The delete marker is used for versioning objects. In COS, the delete marker can be considered as a marker indicating that the object has been deleted. Delete markers can be considered markers with objects that have been deleted. The differences are as follows:

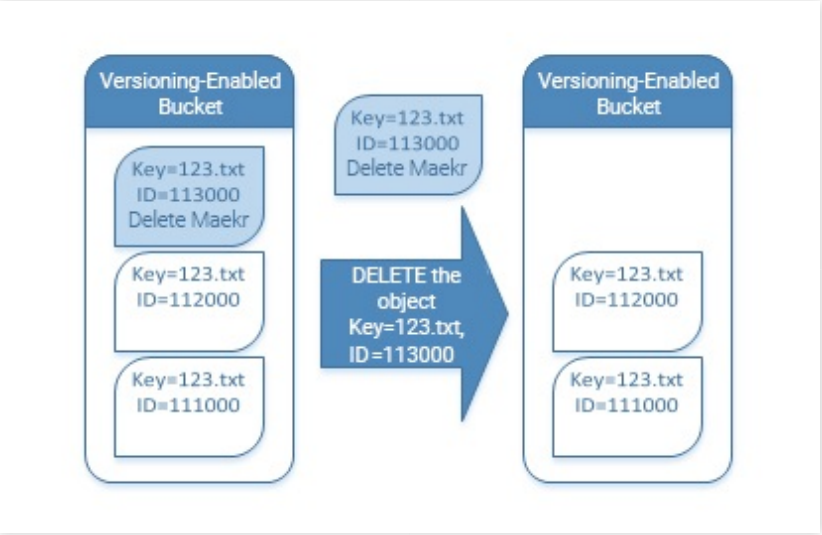
- The content of the delete marker is empty.
- There is no ACL value for the delete marker.
- The delete marker executes a GET request, which returns a 404 error. The response header is x-cos-delete-marker: true .
- The delete marker only supports DELETE operations (requests must be issued under the root account).

Deleting a Delete Marker

If the user needs to delete the delete marker, he can specify its version ID in the DELETE Object versionId request to permanently delete the Delete Marker. If you do not specify the VersionId of a delete marker and issue a DELETE request for a delete marker, COS will not delete the delete marker but will insert a new one. As shown in the following figure, executing a general DELETE request on a delete marker does not delete anything but adds a new delete marker to the bucket.



In a versioning-enabled bucket, newly added delete markers will have a unique VersionId. Therefore, in a bucket, the same object may have multiple delete markers. To permanently delete a Delete Marker, you must include its version ID in the DELETE Object versionId request.
As shown in the following figure, execute the DELETE Object versionId request to permanently delete the Delete Marker.



> Note:
>
> Only the root account can permanently remove the Delete Marker.

Steps to permanently delete the Delete Marker:

1. Set versionId to the versionId of the delete marker.
2. Send a DELETE Object versionId request.

Restoring to an Earlier Version

Versioning can be used to restore an earlier version of an object. There are two ways to do this:

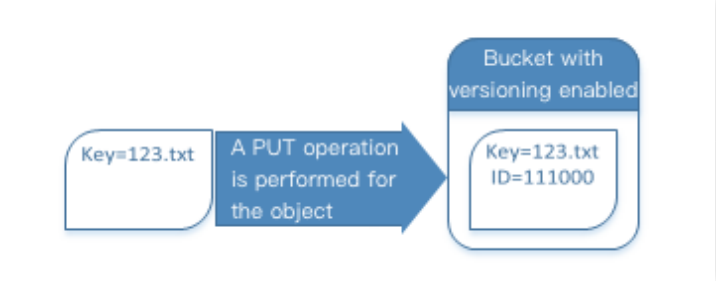
1. Copy an earlier version of an object to the same bucket.
   The copied object becomes the current version of the object, and all object versions are preserved.
2. Permanently delete the current version of an object.
   When you delete the current object version, you actually convert the previous version to the current version of the object.

Managing Objects with Versioning Suspended Status

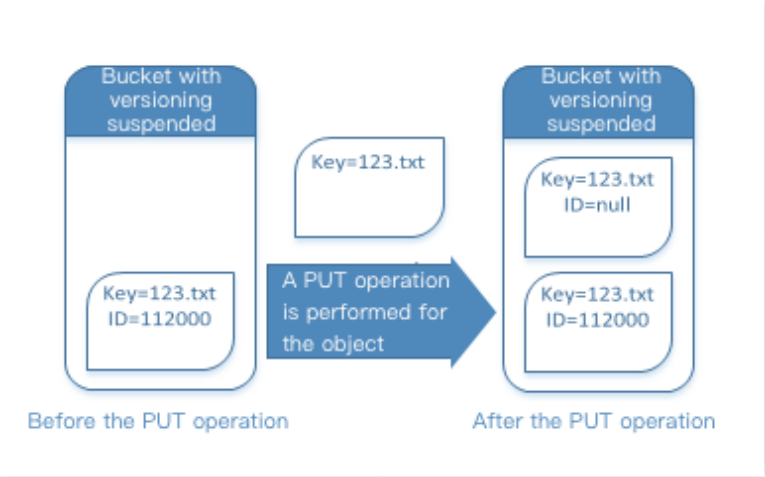
When you suspend versioning, existing objects in the bucket are not changed. What changes is how the COS handles the object in subsequent requests. The following describes how to manage objects in a bucket where versioning has been suspended.

Uploading an Object

After versioning is suspended on a bucket, when a user performs a PUT, POST, or COPY operation, COS automatically adds a Version ID of null to objects placed in the bucket. As shown in the following figure:

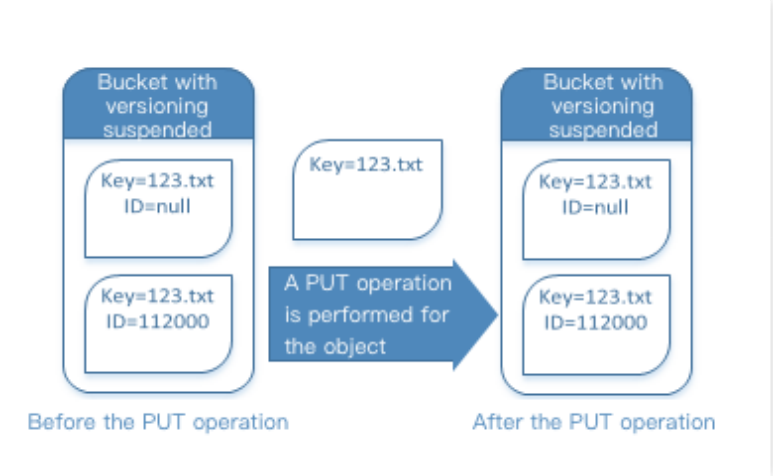


If there are versioning objects in the bucket, the object uploaded to the bucket becomes the current version, and the Version ID is null. As shown in the following figure:



If an empty version already exists in the bucket, it will be overwritten, and the original object content will

be replaced accordingly, as shown in the following figure:



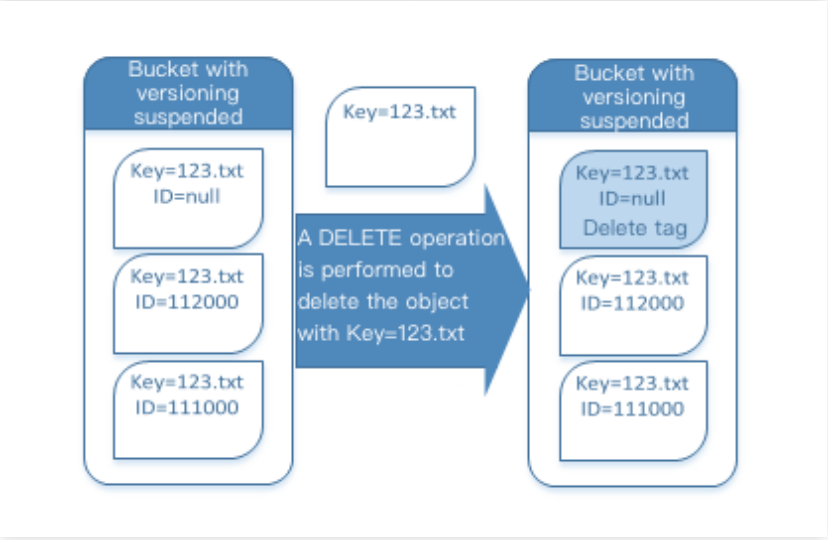
Querying the Metadata Version

On a bucket with versioning suspended, a user issuing a GET Object request will return the current version of the object.

Deleting an Object

If versioning is suspended, the following occurs when executing a DELETE request:

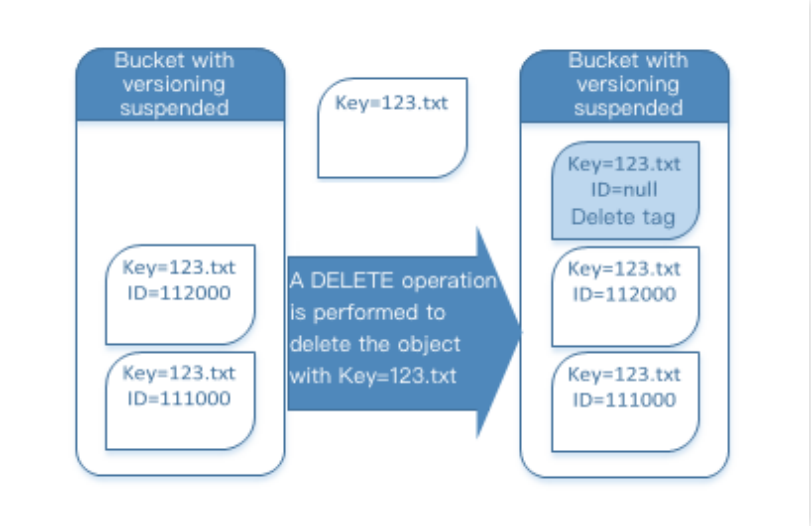
- If there is an empty version of an object in the bucket, the object whose Version ID is null will be deleted.
  As shown in the following figure, when a user performs a general DELETE operation, COS inserts a delete marker for the empty version of the object.



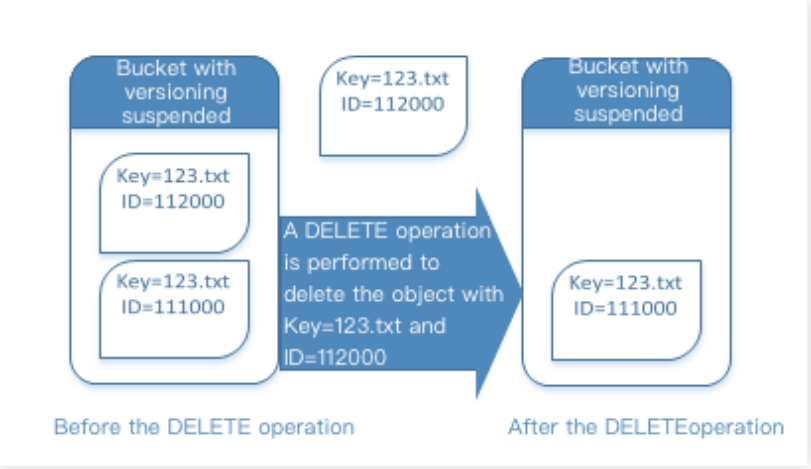
> Note:
>
> The delete marker does not contain any content. When the delete marker replaces the empty

> version, the original content of the empty version will be lost.

- There is no empty version of the object in the bucket, and a delete marker is added to the bucket.
  As shown in the following figure, if there is no empty version in the bucket, the user will not delete anything when performing the DELETE operation. COS only inserts a delete marker.



- Even in a bucket where versioning has been suspended, the root account can permanently delete a specified version.
  As shown in the following figure, deleting a specified object version permanently deletes the object.



> Note:
>
> Only the root account can delete a specified object version.

Deleting Markers

Overview

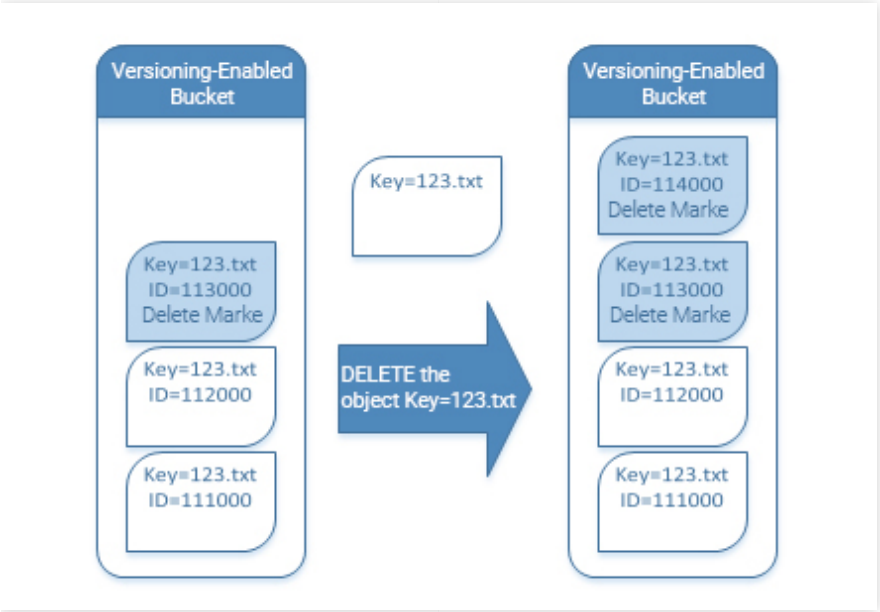
A delete marker is used to mark a versioned object. In COS, a delete marker is considered the marker for a "deleted object". It has an object key and version ID just like an object does, but with the following differences:

- The content of the delete marker is empty.
- There is no ACL value for the delete marker.
- A GET request for the delete marker will return a 404 error.
- The delete marker only supports DELETE operations (requests must be issued under the root account).

Deleting Delete Marker

If the user needs to delete the delete marker, he can specify its version ID in the DELETE Object versionId request to permanently delete the Delete Marker. If you do not specify the VersionId of a delete marker and issue a DELETE request for a delete marker, COS will not delete the delete marker but will insert a new one.

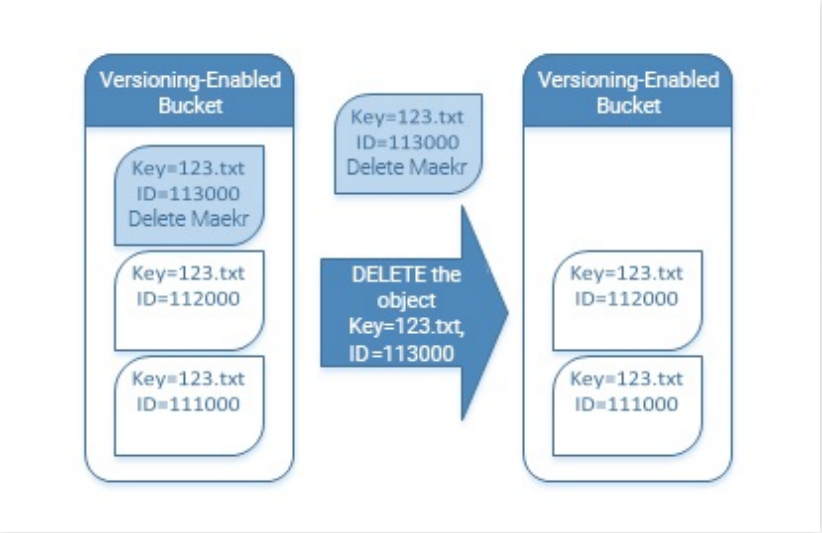
As shown in the following figure, executing a general DELETE request on a delete marker does not delete anything but adds a new delete marker to the bucket.



In a versioning-enabled bucket, newly added delete markers will have a unique VersionId. Therefore, in a

bucket, the same object may have multiple delete markers. To permanently delete a Delete Marker, you must include its version ID in the DELETE Object versionId request.

As shown in the following figure, execute the DELETE Object versionId request to permanently delete the Delete Marker.



> Note:
>
> Only after authorization by the root account DeleteObject operation. The Delete Marker can be deleted only after the operation.

Steps to permanently delete the Delete Marker:

1. Set versionId to the versionId of the delete marker.
2. Send a DELETE Object versionId request.

Versioning Configuration

Use Cases

Versioning allows you to store multiple versions of an object in a bucket and to search for, delete, and restore a specified version of the object.
For more information, see the document Versioning Overview.

> Note:
>
> Only the root account and authorized sub-accounts can configure the versioning status of a bucket.

Usage Instructions

Using the REST API

You can use the REST API to directly configure the versioning of a bucket and manage the objects in a versioning-status bucket. See the following parts in the API documentation:

- PUT Bucket versioning
- GET Buket versioning
- GET Bucket Object versions
- PUT Object
- GET Object
- DELETE Object
- DELETE Multiple Objects

Replication Within the Cloud
Overview of Replication Within the Cloud

Overview

Inter-cloud replication is a configuration for buckets. You can configure inter-cloud replication rules to automatically and asynchronously replication incremental objects from or to buckets in different storage regions. After you enable inter-cloud replication, COS will accurately replicate the object content (information such as the object metadata and version ID) from the source bucket to the destination bucket, and the replication and the original object will have identical attributes. In addition, object operations, such as adding and deleting an object, in the source bucket, will also be replicated to the destination bucket.

> Note:
>
> - To enable the inter-cloud replication feature, you need to ensure that the source and destination buckets are in different storage regions and that the versioning feature is enabled for both buckets.
> - When you enable inter-cloud replication, unless you specify to configure a different storage type for the object replica when you replicate the data, the object replication will be of the same storage type as the source object.
> - During replication, COS will replicate the access control list (ACL) of the source bucket.
> - COS does not support configuring inter-cloud replication rules between two different accounts.

Applicable Scenario

- Remote disaster recovery: COS offers 11 nines 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. If you want to avoid data loss and explicitly have a separate replication of data in a different region, you can use inter-cloud replication to implement inter-cloud disaster recovery. In this way, when an IDC is damaged due to force majeure, the IDC in another region can still provide data copies for your use.
- Compliance requirements: COS offers methods such as multiple copies and erasure coding in physical disks for data by default to ensure data availability. However, some industries may have compliance requirements stipulating that you preserve data copies across different storage regions. Therefore, enabling inter-cloud replication allows you to replicate data across different storage regions to meet such requirements.
- Reducing access latency: When your end users access your objects from different geographic locations, you can use inter-cloud replication to maintain the object copies in the available storage

region closest to your end users' geographic locations. This will minimize your end users' access latency and improve your product experience.
- Special technical requirements: When you have compute clusters in two regions and these clusters need to process the same set of data, you can use inter-cloud replication to maintain object copies in both regions.
- Migrating and backing up data: You can replicate your business data from one available region to another according to the development needs of your business to achieve data migration and backup.

Notes

Limits on Replication Time

The time it takes for COS to replicate an object depends on factors such as the object size, the distance between the storage regions, and the upload method of the object. Based on the above factors, the synchronization time varies from a few minutes to a few hours.

- Object size: It takes more time to replicate larger objects. For them, it is recommended that you use multipart upload to reduce their upload and synchronization time.
- Distance between the storage regions: It takes more data transfer time for a longer distance between the regions.
- Upload method of the object: Simple upload does not support concurrent uploads and can only upload or download data serially over a single connection, but multipart upload supports concurrent uploads. Therefore, using multipart upload for large files will speed up upload and inter-cloud replication.

Notes on Versioning

Inter-cloud replication configuration requires you to configure the versioning feature for both the source and destination buckets. . After you enable versioning, you should note that disabling it will affect the inter-cloud replication feature in the following ways:

- If you try to disable versioning for a bucket where the inter-cloud replication feature is enabled, COS will return an error and prompt you to delete the inter-cloud replication rule before disabling versioning.
- If you try to disable versioning for a destination bucket, COS will prompt you that doing so will affect the inter-cloud replication feature. If you proceed to disable versioning, the inter-cloud replication rule that specifies this bucket as the destination bucket will become invalid in COS.

Replication Actions Description

This document describes what COS will and will not replicate after users enable in-cloud replication for a bucket.

What Will Be Replicated

If you enable the inter-cloud replication feature for a source bucket, COS will replicate the following content:

- Any new objects uploaded to the source bucket after the inter-cloud replication rule is added.
- Object attributes such as an object's metadata and version ID.
- Information on object operations, such as adding an object of the same name (equivalent to adding an object) and deleting an object.
    - If you specify an object version to delete in the source bucket, that is, to specify the version ID, then this delete operation will not be replicated.
    - If you add a bucket-level configuration, such as a lifecycle rule, to the source bucket, then any resulting object operations caused by such a configuration will not be replicated to the destination bucket.

Delete Operations in Inter-Cloud Replication

If you delete an object from a source bucket, the following inter-cloud replication actions will happen:

- For a DELETE request without a specified object version ID, COS will add a delete marker to the source bucket, and the inter-cloud replication feature will replicate this marker to the destination bucket. For more information on versioning and deleting markers, see the document Versioning Configuration.
- For a DELETE request with a specified object version ID, COS will delete the specified object version in the source bucket, but will not replicate this delete operation to the destination bucket. That is, COS will not delete this specified object version in the destination bucket. This action prevents malicious data deletion.

What Will Not Be Replicated

If you enable the inter-cloud replication feature for a source bucket, COS will not replicate the following content:

- The object content that already existed before the inter-cloud replication feature is enabled, that is,

the existing data.

- The encryption information of an encrypted object, namely the encryption information that will be lost after the encrypted object is replicated.
- The object data recently added to the source bucket, if the data is replicated from another bucket.
- The updating actions to bucket-level configuration.
- The results after the execution of lifecycle configuration.
    - The inter-cloud replication of object data among buckets is not transitive. If you set two inter-cloud replication rules at the same time, one of which specifies that bucket A is the source and bucket B is the destination, and the other specifies that bucket B is the source and bucket C is the destination, then the recently added object data in bucket A will only be replicated to bucket B, but not to bucket C.

      Take the lifecycle configuration as an example. After 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. If you want the expired objects to be deleted in the destination bucket as well, you need to configure the same lifecycle rule for the destination bucket additionally.

Configuring Replication Within the Cloud

Applicable Scenario

By configuring the inter-cloud replication rules, you can replicate objects from a source bucket into a specified destination bucket in another region. The inter-cloud replication feature is suitable for scenarios such as implementing inter-cloud disaster recovery, meeting industry compliance requirements, migrating and backing up data, reducing user access latency, and facilitating data access with clusters in different regions.

> Note:
>
> After versioning is enabled, multiple versions will be generated for any recently uploaded objects whose name already exists in the bucket. Each of these versions will occupy your storage capacity and will be billed for storage equally.

Usage Instructions

Using the REST API

You can directly use the REST API to configure and manage the inter-cloud replication rules for a bucket. For details, see the following parts in the API documentation:

- PUT Bucket replication
- GET Bucket replication
- DELETE Bucket replication

Cross-Cloud Replication
Cross-Cloud Replication Overview

Overview

Cross-cloud replication is an advanced feature provided by COS that enables object movement operations between different cloud environments. By configuring cross-cloud replication rules, users can automatically and asynchronously copy incremental objects from the source bucket to the destination bucket.
Once cross-cloud replication is enabled, COS precisely copies the object content in the source bucket, including the object metadata and version ID, ensuring that the copied object replicas have completely consistent cloud data with the source objects. In addition, overwrite operations on objects in the source bucket will be synchronized to the destination bucket. However, it should be noted that deletion operations in the source bucket will not be synchronized through cross-cloud replication.

> Note:
>
> - Cross-cloud replication supports any COS that supports the S3 protocol as the destination.
> - Cross-cloud replication has basically no requirements for the destination bucket. For example, the destination bucket can be a bucket under the same account, a bucket in a different region, or a bucket in a different cloud.
> - Cross-cloud replication supports historical data synchronization, but currently requires manual configuration by the Ops personnel.

Applicable Scenarios

- Implementing cross-region disaster recovery: COS provides 11 nines 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. If you want to avoid data loss and explicitly have a separate replica of data in a different region or cloud, you can use cross-cloud replication to implement cross-region disaster recovery. In this way, when an IDC is damaged due to force majeure, the IDC in another region or cloud can still provide data replicas for your use.
- Meeting compliance requirements: COS provides methods such as multiple replicas and erasure coding in physical disks for data by default to ensure data availability. However, some industries may have compliance requirements stipulating that you preserve data replicas across different storage regions or clouds. Therefore, enabling cross-cloud replication allows you to copy data across different storage regions or clouds to meet such requirements.
- Reducing access latency: When your end users access your objects from different geographic

locations, you can use cross-cloud replication to maintain the object replicas in the available storage region or cloud closest to your end users' geographic locations. This will minimize your end users' access latency and improve your product experience.
- Meeting special technical requirements: When you have compute clusters in two regions or clouds, and these clusters need to process the same set of data, you can use cross-cloud replication to maintain object replicas in both regions.
- Migrating and backing up data: You can copy your business data from one available region or cloud to another according to the development needs of your business to achieve data migration and backup.

Must-Knows

Replication Time Limit

The time it takes for COS to copy an object depends on factors such as the object size, the distance between the storage regions, and the upload method of the object. Based on the above factors, the synchronization time varies from a few minutes to a few hours.

- Object size: It takes more time to copy larger objects. For them, it is recommended that you use multipart upload to reduce their upload and synchronization time.
- Distance between the storage regions or clouds: It takes more data transfer time for a longer distance between the regions or clouds.
- Upload method of the object: A simple upload does not support concurrent uploads and can only upload or download data serially over a single connection, but a multipart upload supports concurrent uploads. Therefore, using multipart upload for large files will speed up upload and cross-cloud replication.

Cross-Cloud Replication Configuration

Applicable Scenarios

By configuring the cross-cloud replication rules, users can copy objects from a source bucket into a specified destination bucket. The cross-cloud replication feature is suitable for scenarios such as implementing cross-region disaster recovery, meeting industry compliance requirements, migrating and backing up data, reducing user access latency, and facilitating data access with clusters in different regions or clouds.

Usage Method

Using the Tenant-Side Interface

You can directly log in to the tenant side and access the bucket interface to configure cross-cloud replication rules.

Using the REST API

You can directly use the REST API to configure and manage the in-cloud replication rules for a bucket. For details, see the following parts in the API documentation:

- Put Bucket CrossCloudReplication
- GET Bucket CrossCloudReplication
- DELETE Bucket CrossCloudReplication

Data Security
Server-Side Encryption Overview

Overview

COS encrypts your data at the object level before the data is written to the disks in the Internet Data Center (IDC) and automatically decrypts the data when you access it. Encryption and decryption are completed on servers. This server-side encryption feature can effectively protect static data.

> Note:
>
> - There is no difference in experience between accessing an encrypted object and accessing an unencrypted object, provided that you have access to the objects.
> - 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.
> - Objects encrypted with SSE-KMS cannot be accessed by anonymous users.

Applicable Scenario

- 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 Methods

COS supports multiple server-side encryption methods such as SSE-COS, SSE-KMS and SSE-C. You can choose a suitable one to encrypt your data stored in COS.

SSE-COS Encryption

SSE-COS encryption is server-side encryption with a key managed by COS. In this mode, COS manages the primary key and the data, and you can manage and encrypt your data directly through COS. SSE-COS adopts strong multi-factor encryption, ensuring that each object is encrypted with a unique key. At the same time, it adopts the Advanced Encryption Standard with a 256-bit key (AES-256) for data encryption,

and regularly rotates the primary 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.
> - If you upload an object with a pre-signed URL, you cannot use SSE-COS encryption to encrypt it. You can only use the COS console or HTTP request header to specify server-side encryption.

Using the COS Console

For details on how to perform SSE-COS encryption on objects via the console, see the document Setting Object Encryption.

Using the REST API

> 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 providing the request header.

When you request the following APIs, you can apply server-side encryption by providing the x-cos-server-side-encryption header.

- PUT Object
- Initiate Multipart Upload
- PUT Object - Copy
- POST Object

SSE-KMS Encryption

SSE-KMS encryption is server-side encryption with a key managed by KMS. KMS is a security management service launched by Converge Cloud and uses a third-party-certified Hardware Security Module (HSM) to generate and protect keys. KMS allows you to easily create and manage keys, meeting your 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 in the KMS console, and define key policies and use methods. KMS allows you to choose your own key material from KMS or External sources.

> Note:

> - SSE-KMS only encrypts the object data, not its metadata.
> - Using SSE-KMS encryption will incur an additional cost, which will be charged by KMS.
> - Objects encrypted with SSE-KMS can only be accessed with a valid signature but not by anonymous users.

Using the COS Console

For details on how to perform SSE-KMS encryption on objects via the console, see the document Setting Object Encryption.

Using the REST API

> 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 providing the request header.

When you request the following APIs, you can apply server-side encryption by providing the x-cos-server-side-encryption header.

- PUT Object
- Initiate Multipart Upload
- PUT Object - Copy
- POST Object

Notes

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. The specific steps are shown below:

1. Log in to the CAM console and go to the role management list page.
2. Click Create Role, enter the corresponding parameters as prompted on the interface, and then click Next: Configure role policy.
3. Configure the role policy, search for and check KMSAccessForCOSRole, and click Next: Review.
4. Click Complete to complete the process.

SSE-C Encryption

SSE-C encryption is server-side encryption with a user-defined key, so you need to provide the encryption 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. This value is used to verify your request to access the object. COS cannot use the HMAC value with random data to derive the value of the encryption key or to decrypt the encrypted object. Therefore, if you lose the encryption key, you cannot get the object again.
> - When you use the POST operation to upload an object, instead of providing the x-cos-server-side-encryption-* header, you need to provide the same information in the form field.
> - You can only use SSE-C with APIs but not console operations.

Using the REST API

When you request the following APIs, in terms of PUT and POST requests, you can provide the x-cos-server-side-encryption-* header to apply server-side encryption; in terms of GET and HEAD requests for the use of an SSE-C encrypted object, you need to provide the x-cos-server-side-encryption-* header to decrypt the specified object. For more information, see Common Request Headers - SSE-C. This header is supported by the following operations:

- GET Object
- HEAD Object
- PUT Object
- Initiate Multipart Upload
- Upload Part
- POST Object
- PUT Object - Copy

Bucket Encryption Overview

Overview

Bucket encryption is a configuration for a bucket. By setting bucket encryption, you can encrypt all new objects uploaded to a bucket with the specified encryption method by default.

Currently the following encryption methods are supported:

- SSE-COS encryption: Server-side encryption with a key managed by COS.
- SSE-KMS encryption: Server-side encryption with a key managed by KMS.

Usage Instructions

Using the COS Console

You can use the COS console to set bucket encryption.

Using the REST API

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.

Access Management
Access Policy Language Overview

Overview

Access policies use the JSON-based access policy language and are used to grant access to COS resources. You can use the access policy language to authorize a specified principal to execute specified operations on specified COS resources.

Access Policy Elements

The access policy language contains the following elements with basic meanings:

- Principal: Describes the entity to be authorized by the policy. This includes entities such as users (root accounts, sub-accounts, anonymous users) and user groups. This element is valid for bucket access policies but cannot be added for 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, and resource. A policy can and has only one statement element.
    - Effect: Describes whether the result of the statement is an "allow" or an explicit "deny". This includes allow and deny. This element is required.
    - Action: Describes the allowed or denied action. An action can be an API (described with the prefix "name") or a feature set (a set of specific APIs described with the prefix "permid"). This element is required.
    - Resource: Describes the detailed data authorized. 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 product documentation corresponding to the resources you are writing a statement for. This element is required.

Element Usage

Specifying a Principal

The principal element is used to specify the user, account, service, or another entity that is allowed or denied to access resources. This element only works in buckets; it doesn't need to be specified in user policies because such policies are directly added to the specific user. A sample specifying a principal is

shown below.

```
"principal": {
 "qcs": [
  "qcs::cam::uin/1:uin/1"
 ]
}
```

Granting permissions to anonymous users:

```
"principal": {
 "qcs": [
  "qcs::cam::anonymous:anonymous"
 ]
}
```

Granting permissions to the root account with UIN 1200000313:

```
"principal": {
 "qcs": [
  "qcs::cam::uin/1200000313:uin/1200000313"
 ]
}
```

Grant permissions to a sub-account UIN 3030313 (under the root account UIN 1200000313):

> Note:
>
> Before the action, ensure that the sub-account has been added to the sub-account list of the root account.

```
"principal": {
 "qcs": [
  "qcs::cam::uin/1200000313:uin/3030313"
 ]
}
```

Specifying an Effect

If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to (deny) a resource. In this way, you can ensure that users cannot access the resource, even if a different policy grants access. A sample specifying an allowed effect is shown below.

```
"effect" : "allow"
```

Specifying an Action

COS defines a specific COS action that you can specify in a policy. The specified action is identical to the initiated API request operation.

Bucket Operations

Description	Corresponding API
name/cos:GetService	GET Service
name/cos:GetBucket	GET Bucket (List Object)
name/cos:PutBucket	PUT Bucket
name/cos:DeleteBucket	DELETE Bucket
name/cos:HeadBucket	HEAD Bucket
name/cos:GetBucketPolicy	GET Bucket Policy
name/cos:PutBucketPolicy	PUT Bucket Policy
name/cos:DeleteBucketPolicy	DELETE Bucket Policy
name/cos:GetBucketACL	GET Bucket ACL
name/cos:PutBucketACL	PUT Bucket ACL
name/cos:ListMultipartUploads	LIST in-progress multipart uploads

Object Operations

Description	Corresponding API
name/cos:GetObject	GET Object
name/cos:PutObject	PUT Object
name/cos:HeadObject	HEAD Object
name/cos:DeleteObject	DELETE Object
name/cos:PutObjectCopy	COPY Object
name/cos:PostObject	POST Object
name/cos:GetObjectACL	GET Object ACL
name/cos:PutObjectACL	PUT Object ACL

Description	Corresponding API
name/cos:InitiateMultipartUpload	Initiate a multipart upload
name/cos:UploadPart	Upload a part in a multipart upload
name/cos:CompleteMultipartUpload	Complete a multipart upload
name/cos:AbortMultipartUpload	Abort a multipart upload

A sample specifying an action that is allowed is shown below:

```
"action": [
  "name/cos:GetObject",
  "name/cos:HeadObject"
]
```

Specifying a Resource

The resource element describes one or multiple operation objects, such as COS buckets and objects. All resources can be described in the following six-segment format.

```
qcs:project_id:service_type:region:account:resource
```

Where:

- qcs is the abbreviation of qcloud service.
- project_id describes project information, only for compatibility with early CAM logic. Please leave this blank here.
- service_type describes the abbreviation of the product, such as COS.
- region describes the region information.
- account describes the root account information of the resource owner.
- resource describes the detailed resource information. In COS services, it is described using the bucket XML API access endpoints.

A sample specifying the bucket burningtest-1251500699 is shown below.

```
"resource": ["qcs::cos:ap-city:uid/1251500699:burningtest-1251500699/*"]
```

A sample specifying all objects in the /test/ folder in the bucket burningtest-1251500699 is shown below.

```
"resource": ["qcs::cos:ap-city:uid/1251500699:burningtest-1251500699/test/*"]
```

A sample specifying the /test/1.txt object in the bucket burningtest-1251500699 is shown below.

"resource": ["qcs::cos:ap-city:uid/1251500699:burningtest-1251500699/test/1.txt"]

Practical Case

When the access source IP addresses are 101.226.100.185 and 101.226.100.186, if the root account allows anonymous users to execute the GET (download) and HEAD actions on objects in the bucket burningtest-1251500699 in city region, no authentication is required.

```
{
  "version": "2.0",
  "principal": {
    "qcs": [
      "qcs::cam::anonymous:anonymous"
    ]
  },
   "statement": [
     {
        "action": [
           "name/cos:GetObject",
           "name/cos:HeadObject"
        ],
        "effect": "allow",
        "resource": [
           "qcs::cos:city:uid/1251500699:burningtest-1251500699/*"
        ]
     }
   ]
}
```

Best Practices
Access Control and Permission Management
ACL Access Control Practice

Overview

An access control list (ACL) is one of the resource-based access policy options that manages access to buckets and objects. An ACL can be used to grant read and write permissions to other root accounts and user groups.

Unlike an access policy, an ACL has the following limits on the management permissions:

- Permissions can only be granted to accounts of the cloud platform.
- Only supports five operation groups: Read Objects, Write to Objects, Read ACLs, Write to ACLs, and All Permissions.
- Additional conditions for the ACL rules to take effect are not supported.
- Explicit deny is not supported.

Control granularities supported by an ACL:

- Bucket
- Object key prefix
- Object

Control Elements of an ACL

When a bucket or an object is created, the root account to which the resources of the bucket or object belong owns all permissions to these resources, and the permissions cannot be modified or deleted. You can use an ACL to grant other cloud platform accounts the access permissions. The following is an example ACL for a bucket.

```
<AccessControlPolicy>
  <Owner>
    <ID>qcs::cam::uin/12345:uin/12345</ID>
    <DisplayName>qcs::cam::uin/12345:uin/12345</DisplayName>
  </Owner>
```

```
<AccessControlList>
  <Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="RootAccount">
      <ID>qcs::cam::uin/12345:uin/12345</ID>
      <DisplayName>qcs::cam::uin/12345:uin/12345</DisplayName>
    </Grantee>
    <Permission>FULL_CONTROL</Permission>
  </Grant>
  <Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="RootAccount">
      <ID>qcs::cam::uin/54321:uin/54321</ID>
      <DisplayName>qcs::cam::anyone:anyone</DisplayName>
    </Grantee>
    <Permission>READ</Permission>
  </Grant>
</AccessControlList>
</AccessControlPolicy>
```

The preceding ACL contains an Owner element that identifies the bucket owner who owns all permissions to the bucket, and the Grant element is used to grant anonymous read permission, which is expressed as READ permission of qcs::cam::anyone:anyone .

Grantees

Root Account

You can grant access permissions to other root accounts based on the definition of the principal in CAM, which is described as follows:

```
qcs::cam::uin/1238423:uin/1238423
```

Anonymous User

You can grant access permissions to anonymous users based on the definition of the principal in CAM, which is described as follows:

```
qcs::cam::anyone:anyone
```

Permission Operation Group

The following table describes the permission operation groups supported by an ACL.

Operation Group	Bucket	Prefix	Object
READ	List and read objects in a	List and read objects in a	Read the Object

Operation Group	Bucket	Prefix	Object
	bucket.	directory.	
WRITE	Create, overwrite, or delete any object in a bucket.	Create, overwrite, or delete any object in the directory.	Overwrite and Delete the Object
READ_ACP	Read the ACL of a bucket.	Read the ACL of a directory.	Read an object's ACL.
WRITE_ACP	Modify the ACL of a bucket.	Modify the ACL under a directory.	Modify the ACL of an object.
FULL_CONTROL	Any operation on buckets and objects.	Perform any operation on the object in the directory.	Perform any operation on the object.

Standard ACL Description

COS supports a series of predefined authorizations, which are called standard ACLs. The following table describes the authorization meanings of standard ACLs.

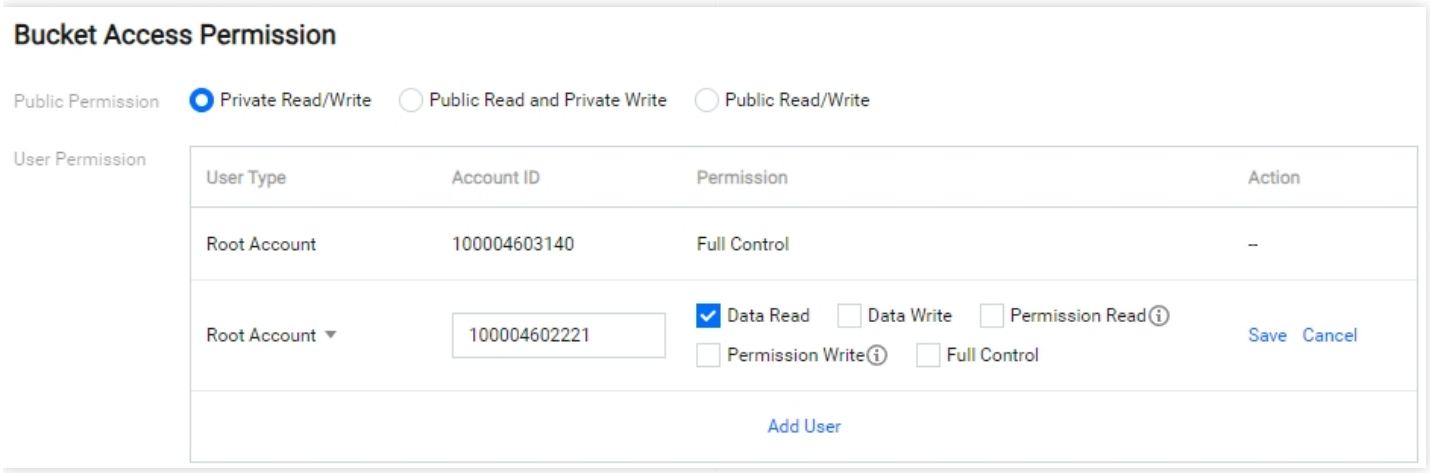
> Note:
>
> A root account always has the FULL_CONTROL permission, which is not described in the following table.

Standard ACL	Description
(Null)	This is the default policy. Others do not have permissions. The permissions for resources are inherited from the upper level.
private	Others do not have permissions.
public-read	The anonymous user group has the READ permission.
public-read-write	The anonymous user group has the READ and WRITE permissions, which are generally not recommended for buckets.

ACL Examples

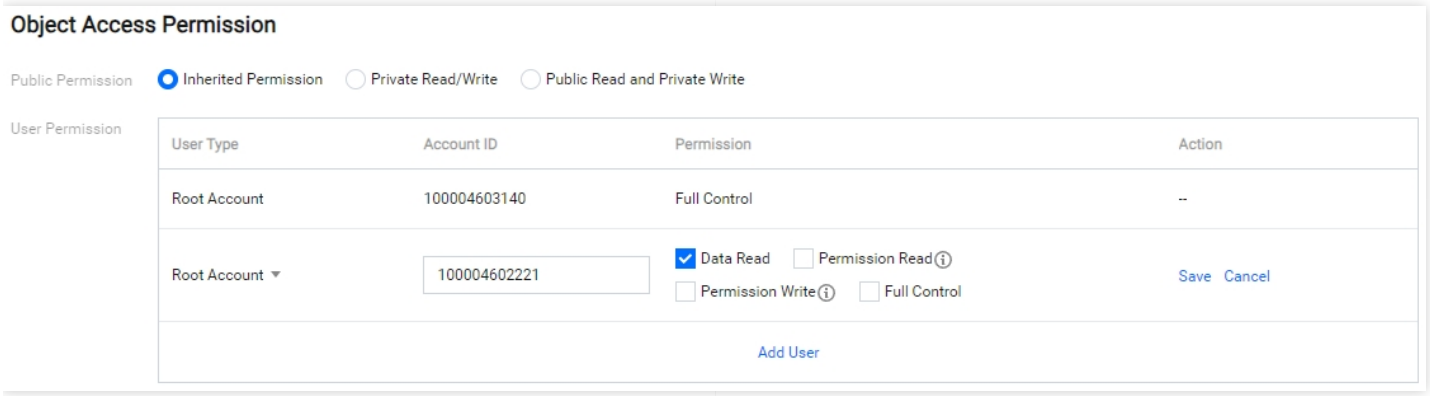
Setting an ACL for a Bucket

The following example grants another root account the read permission to a bucket.



Setting an ACL for an Object

The following example grants another root account the read permission to an object.



CAM Practices

Overview

Cloud Access Management (CAM) is an identity authentication and authorization service provided by the cloud platform. It helps customers manage the access permissions to resources under cloud platform accounts. When permissions are granted, it manages authorized objects, resources, and operations and sets policy limits.

Features

Authorizing Permissions to Root Account Resources

You can grant the access to resources under a root account to other users, including sub-accounts and other root accounts, without sharing the identity credentials of the root account.

Refined Permission Management

Different access permissions for different resources can be granted to different users. For example, some sub-accounts can be granted the read permission to a COS bucket, and some other sub-accounts or root accounts can be granted the write permission to a COS object. Such resources, access permissions, and users can be managed in batch.

Use Cases

Enterprise Sub-Account Permission Management

Employees in different positions in an enterprise need to have minimized access permissions to the enterprise's cloud resources.

Scenario: An enterprise owns many cloud resources, including CVM, VPC instances, CDN instances, COS buckets, and objects, and has many employees, including developers, testers, and Ops engineers. Developers and testers need the read and write permissions of project-specific cloud resources on development servers and test servers, respectively, and Ops engineers are responsible for the purchase and routine operations of such servers. If the duty or project of an employee changes, the corresponding permissions should be terminated.

Cross-Enterprise Permission Management

There are cases where enterprises may need to share their cloud resources.

Scenario: Enterprise A has many cloud resources and wants to focus on product R&D, so it outsources the Ops of its cloud resources to enterprise B, and it needs to revoke all the granted permissions as soon as the outsourcing service contract is terminated.

Policy Syntax

A CAM policy consists of several elements and is used to describe specific information on authorization. Core elements include principal, action, resource, condition, and effect.

Policy Syntax Description

- There is no particular sequence in the description of policy syntax. However, note that the elements are case-sensitive.
- If there are no particular conditions required, the condition element is optional.
- You can define the principal element only through policy management APIs or policy syntax parameters but not in the console.

Version

Specifies the version of the policy syntax. Valid value: 2.0. This element is required.

Principal

Describes the entity to be authorized by the policy, including users (developers, sub-accounts, anonymous users) and user groups. This element can be used only in policy management APIs and policy syntax parameters.

Statement

Describes the detailed information of one or more permissions. This element includes permissions or permission sets of other elements, such as action, resource, condition, and effect. A policy has one and only one statement element. This element is mandatory.

Action

Describes the action to be allowed or denied. Operations can be APIs (described with a name prefix) or feature sets (a specific set of APIs described with a permid prefix). This element is required.

Resource

Describes the specific data authorized. Resources are described in six segments. The details of resource definitions for each product may vary. For information on specifying resources, see the product documentation corresponding to the resource statement you are writing. This element is mandatory.

Condition

Describes the conditions under which the policy takes effect. Conditions consist of operators, keys, and values. Condition values can include information such as time and IP addresses. Some services allow you to specify additional values in conditions. This element is optional.

Effect

Describes whether the statement result is allow or deny. This element is mandatory.

Policy Limits

Restriction Item	Limit
Number of user groups under a root account	20
Number of sub-users under a root account	1000
Number of user groups that a sub-user can be added to	10
Number of sub-users in a user group	100
Number of policies under a root account	1000
Number of policies that can be directly associated with a user or user group	20
Number of characters in a custom policy	4,096 characters

Authorizing a Sub-account to Access COS

Overview

Different access permissions to COS resources need to be configured for different teams or users in different enterprises or departments. You can set different operation permissions on buckets or objects through CAM, so that different teams or users in different companies or departments can collaborate with each other.

First, you need to understand several terms: root account, sub-account (user), and user group. For CAM terms and detailed configuration description, see CAM.

Root Account

A root account is also known as a developer. When a user applies for an account, the system creates a root account for logging in to services. A root account is the primary entity for metering and billing resource usage. By default, a root account owns all permissions to all owned resources, including accessing billing information, modifying the user password, creating users and user groups, and accessing other cloud service resources. By default, only the root account can access resources, and other users need to be authorized by the root account to access resources.

Sub-Accounts (Users) and User Groups

- A sub-account is an entity created by a root account that has a definite ID and login credential and can log in to the Converge Cloud console.
- Sub-accounts do not own resources by default and must be authorized by the root account to which they belong.
- One root account can create multiple sub-accounts (users).
- A sub-account can belong to multiple root accounts, and assist multiple root accounts in managing their respective cloud resources. However, a sub-account can only log in to one root account at a time to manage the cloud resources of the root account.
- Sub-accounts can switch developers (root accounts) through the console, and switch from one root account to another.
- When a sub-account logs in to the console, it will automatically switch to the default root account and have the access permissions granted by the default root account.
- After switching developers, the sub-account will have the access permissions authorized by the root account it switches to, and the access permissions granted by the root account before the switch will become invalid immediately.
- A user group is a collection of multiple users (sub-accounts) with the same function. You can create different user groups based on business needs and associate appropriate policies with user groups to assign different permissions.

Directions

Authorizing a sub-account to access COS consists of three steps: creating a sub-account, granting permissions to the sub-account, and accessing COS resources with the sub-account.

Step 1: Create a Sub-Account

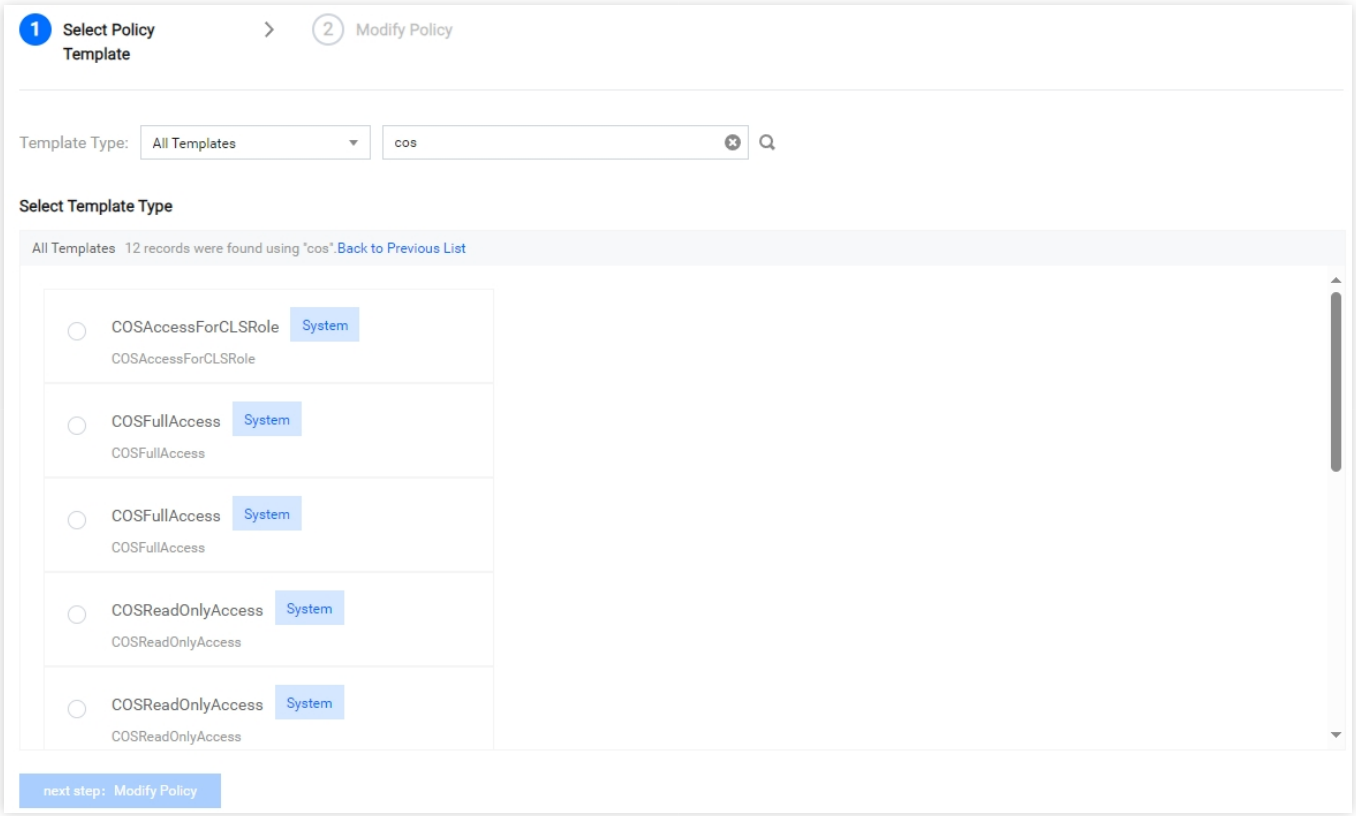
In the CAM console, you can create sub-accounts and configure the access permissions granted to the sub-accounts. The specific operations are as follows:

1. Log in to the CAM console.
2. In the left navigation tree, choose User Management > User to enter the User Management page.
3. Click Create User and select a Sub-user type to go to the Create Sub-user page.
4. Enter user information as required and click Create.
5. Set permissions for the sub-user. Based on the policies provided by the system, you can configure simple policies, such as the access or read-only permission to the COS bucket list.
   To configure a more complex policy, proceed to Step 2: Grant permissions to sub-accounts.
6. After confirming that the entered user information is correct, click Complete.

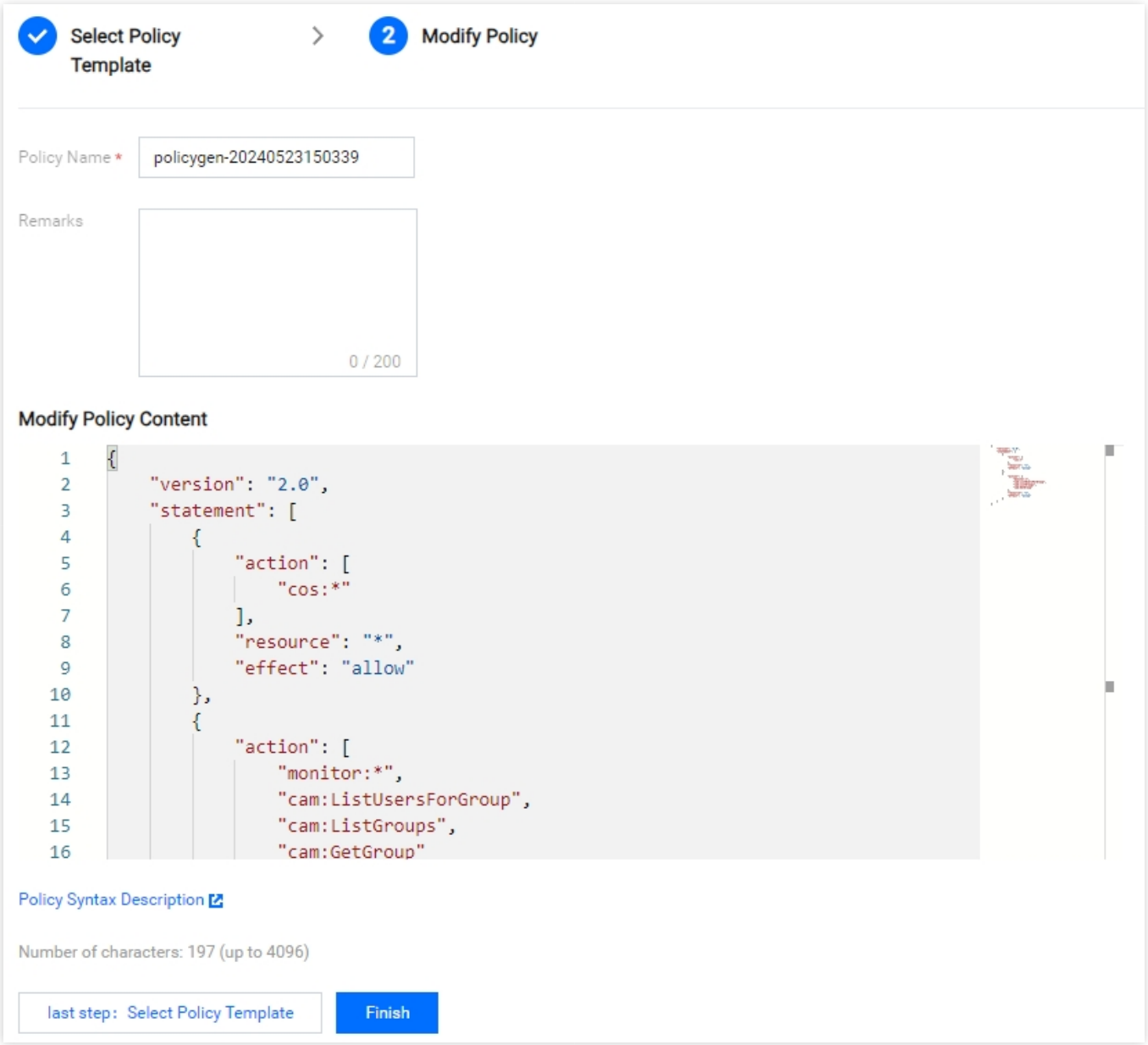
Step 2: Grant Permissions to the Sub-Account

To grant permissions to sub-accounts, you can use CAM to configure policies for sub-accounts (users) or user groups.

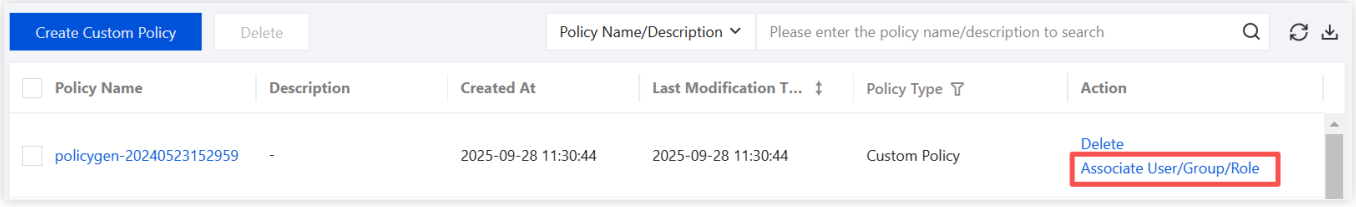
1. Log in to the CAM console.
2. Choose Policy Management > Create Custom Policy > Create by Policy Syntax to enter the policy creation page.
3. Available templates include the Blank Template and the Predefined Policy Template associated with COS. Select the policy template you want to grant to the sub-account and click next step: Modify Policy.



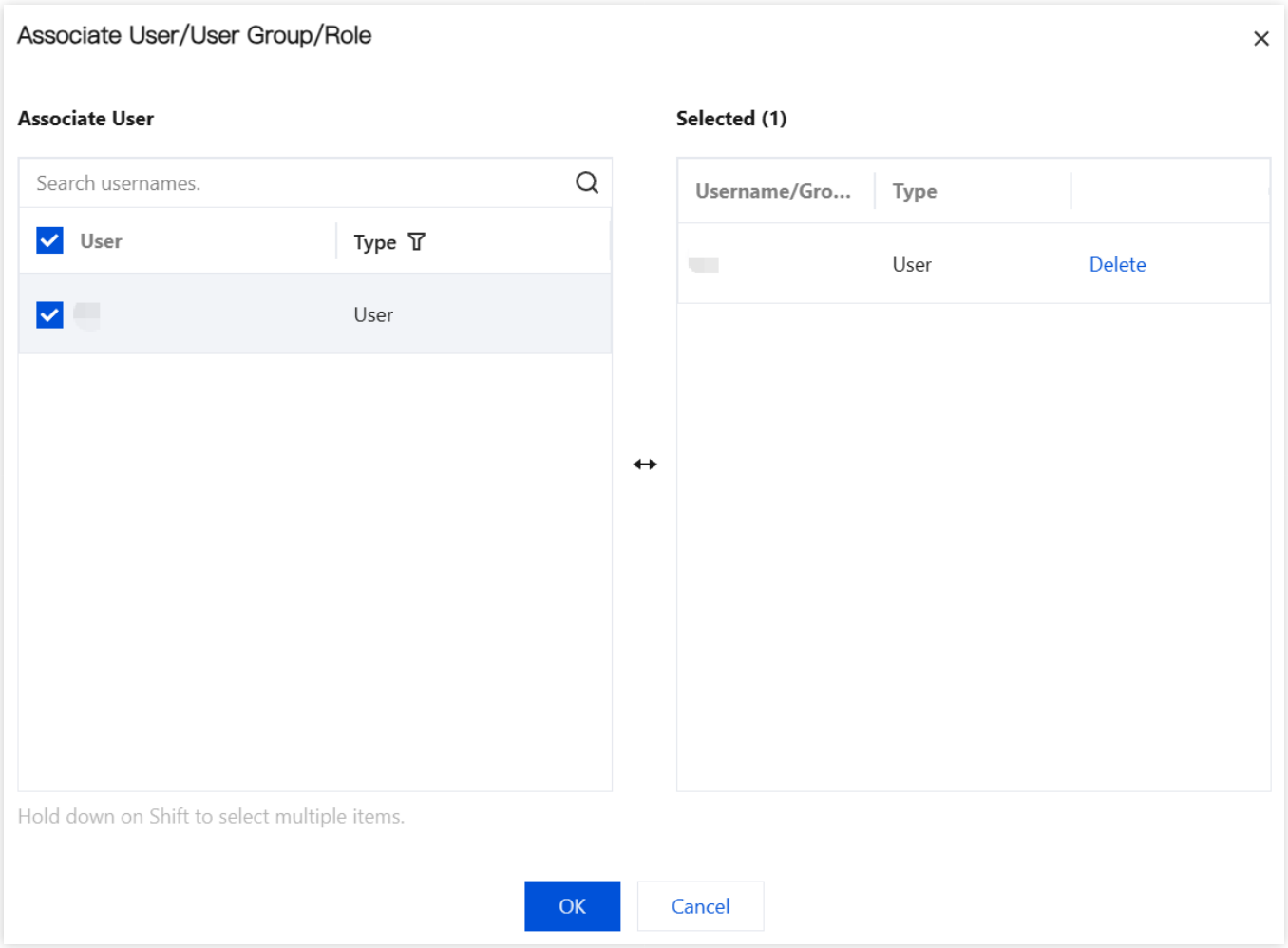
4. Enter a policy name that is easy for you to remember. If you select the Blank Template, you need to enter your policy syntax. For details, see Policy Example. You can copy and paste the policy content into the Modify Policy Content input box, and click Finish.



5. After the policy is created, associate it with the sub-account.



6. After you select the sub-account to confirm authorization, the sub-account can access COS resources within the permission scope.



Step 3: Accessing COS Resources with the Sub-Account

To access COS (API or SDK), the following resources are required: APPID, SecretId, and SecretKey.
When you access COS resources with a sub-account, you need to use the app ID of the root account and the SecretId and SecretKey of the sub-account. You can generate the SecretId and SecretKey of the sub-account in the CAM console.

1. Log in to the CAM console with the root account.
2. In the left navigation tree, choose User Management > User to enter the User Management page.
3. Click the sub-account user name to enter the sub-account information page.
4. Select the API Key Management tab and click Create Key to create a SecretId and SecretKey for the sub-account.
   In this way, you can access COS resources with the SecretId and SecretKey of the sub-account and the app ID of the root account.

   > Note:
   >
   > - The sub-account needs to access COS resources by using the XML API or an SDK based on the XML API.
   > - When a sub-account accesses COS resources, it needs to use the app ID of the root account and

> the SecretId and SecretKey of the sub-account.

XML-based Java SDK Access Example

Taking the XML-based Java SDK command line as an example, the parameters to be filled in are as follows:

```
// 1 Initialize identity information
COSCredentials cred = new BasicCOSCredentials("<root account APPID>", "<sub-account SecretId>", "<sub-account SecretKey>");
```

Example:

```
// 1 Initialize identity information
COSCredentials cred = new BasicCOSCredentials("1250000000", "AKIDasdfmRxHPa9oLhJp", "e8Sdeasdfas2238Vi");
```

Example of Access Through COSCMD Command Line Tool

Taking COSCMD config command line as an example, the parameters to be filled in are as follows:

```
coscmd config -u <root account APPID> -a <sub-account SecretId> -s <sub-account SecretKey> -b <root account bucketname> -r <root account bucket region>
```

Example:

```
coscmd config -u 1250000000 -a AKIDasdfmRxHPa9oLhJp -s e8Sdeasdfas2238Vi -b examplebucket -r ap-city
```

Policy Examples

The following provides policy examples in several typical scenarios. When configuring a custom policy, you can copy and paste the following reference policies into the Modify Policy Content input box and modify them based on the actual configuration.

Configuring Read/Write Permissions for Sub-Accounts

Only read/write permissions are configured for the sub-account. The specific policy is as follows:

```
{
    "version": "2.0",
    "statement": [
        {
            "action": [
                "name/cos:*"
            ],
            "resource": "*",
```

```
            "effect": "allow"
        },
        {
            "effect": "allow",
            "action": "monitor:*",
            "resource": "*"
        }
    ]
}
```

Configuring Read-Only Permissions for Sub-Accounts

Configure read-only permissions for the sub-account. The specific policy is as follows:

```
{
    "version": "2.0",
    "statement": [
        {
            "action": [
                "name/cos:List*",
                "name/cos:Get*",
                "name/cos:Head*",
                "name/cos:OptionsObject"
            ],
            "resource": "*",
            "effect": "allow"
        },
        {
            "effect": "allow",
            "action": "monitor:*",
            "resource": "*"
        }
    ]
}
```

Configuring Read/Write Permissions for a Certain IP Range for a Sub-Account

In this example, only IP addresses in the 192.168.1.0/24 and 192.168.2.0/24 IP ranges have read and write permissions, as shown in the following.
For detailed effective conditions, see Effective Conditions.

```
{
    "version": "2.0",
    "statement": [
        {
            "action": [
    "cos:*"
            ],
```

```
            "resource": "*",
            "effect": "allow",
            "condition": {
                "ip_equal": {
                    "qcs:ip": ["192.168.1.0/24", "192.168.2.0/24"]
                }
            }
        }
    ]
}
```

Cases Related to Permission Settings

Cases for Authorization Through a Bucket Policy

Preparations

1. Create a bucket.
   Authorization through a bucket policy applies only to specific buckets, and you need to create a bucket first. If account-level authorization is required, see Cases Related to Permission Settings in this document.

2. Prepare the UIN of the account to be authorized.
   It is assumed that the UIN of the root account owning the target bucket is 100000000001, and the UIN of a sub-account under it is 100000000011. The sub-account needs to be authorized to access the target bucket.

   > Note:
   >
   > - To query sub-accounts created under the root account, log in to the CAM console and view them in the user list.
   > - To create a sub-account, see Creating a Sub-User.

3. Open the Add Policy dialog box.
   Go to Permission Management of the target bucket, select Policy Permission Settings > Graphical Settings, and click the target bucket name to open the Add Policy dialog box. Then, configure the policy based on the authorization cases in this document. For detailed guidance on how to add a policy, see Adding a Bucket Policy.

The following lists several different authorization cases, and you can configure data based on the actual situation.

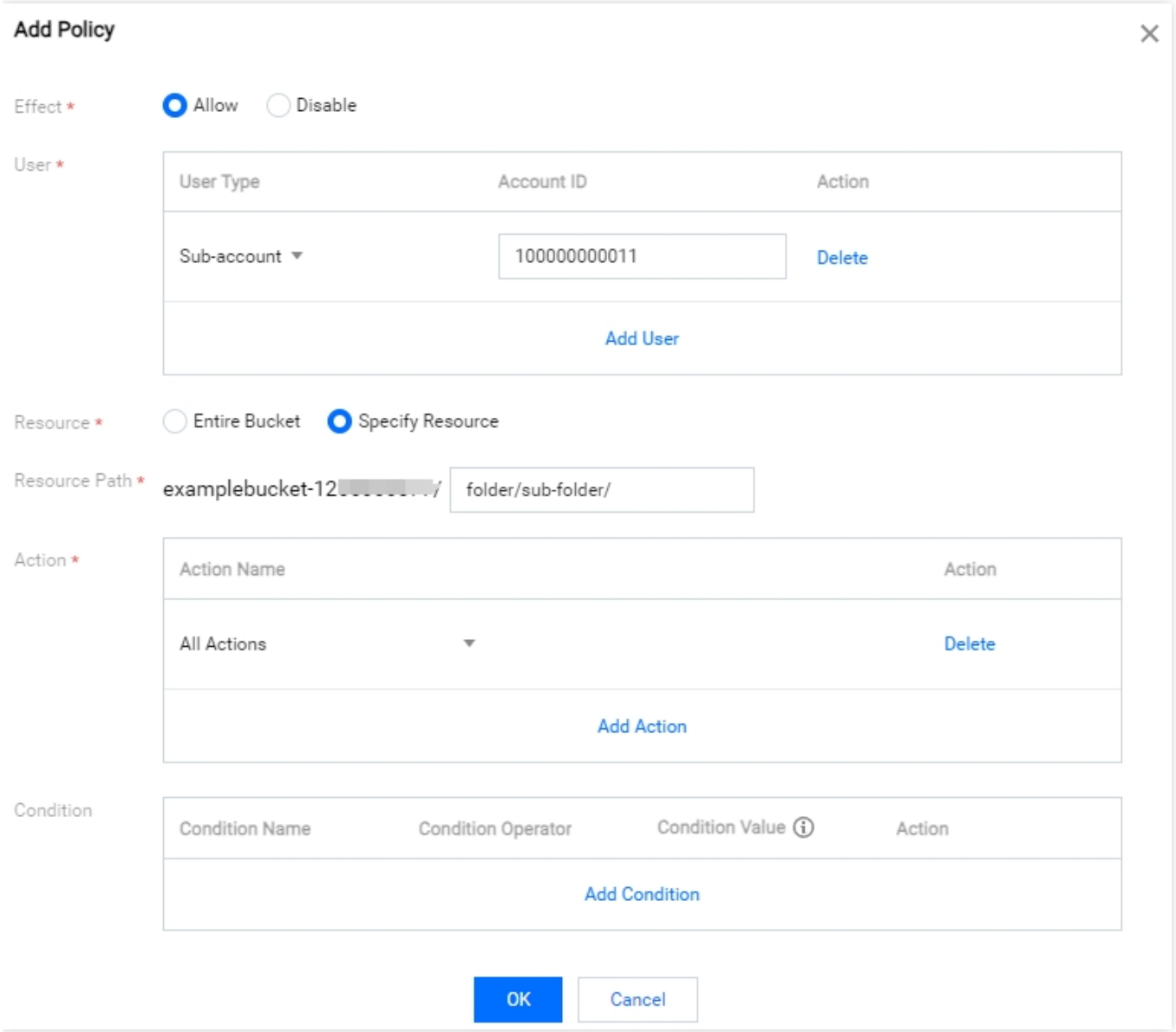
Authorization Cases

Case 1: Authorizing a Sub-account All Permissions to a Specific Directory

The following table describes the configuration information.

Configuration Item	Configuration Value
Effect	Allow
User Type	Sub-account
Account ID.	UIN of a sub-account, which should be a sub-account under the root account, for example, 100000000011

Configuration Item	Configuration Value
Resources	Specified resource
Resource path	Specified directory prefix, such as folder/sub-folder/ , which should be ended with /
Operation name	All operations

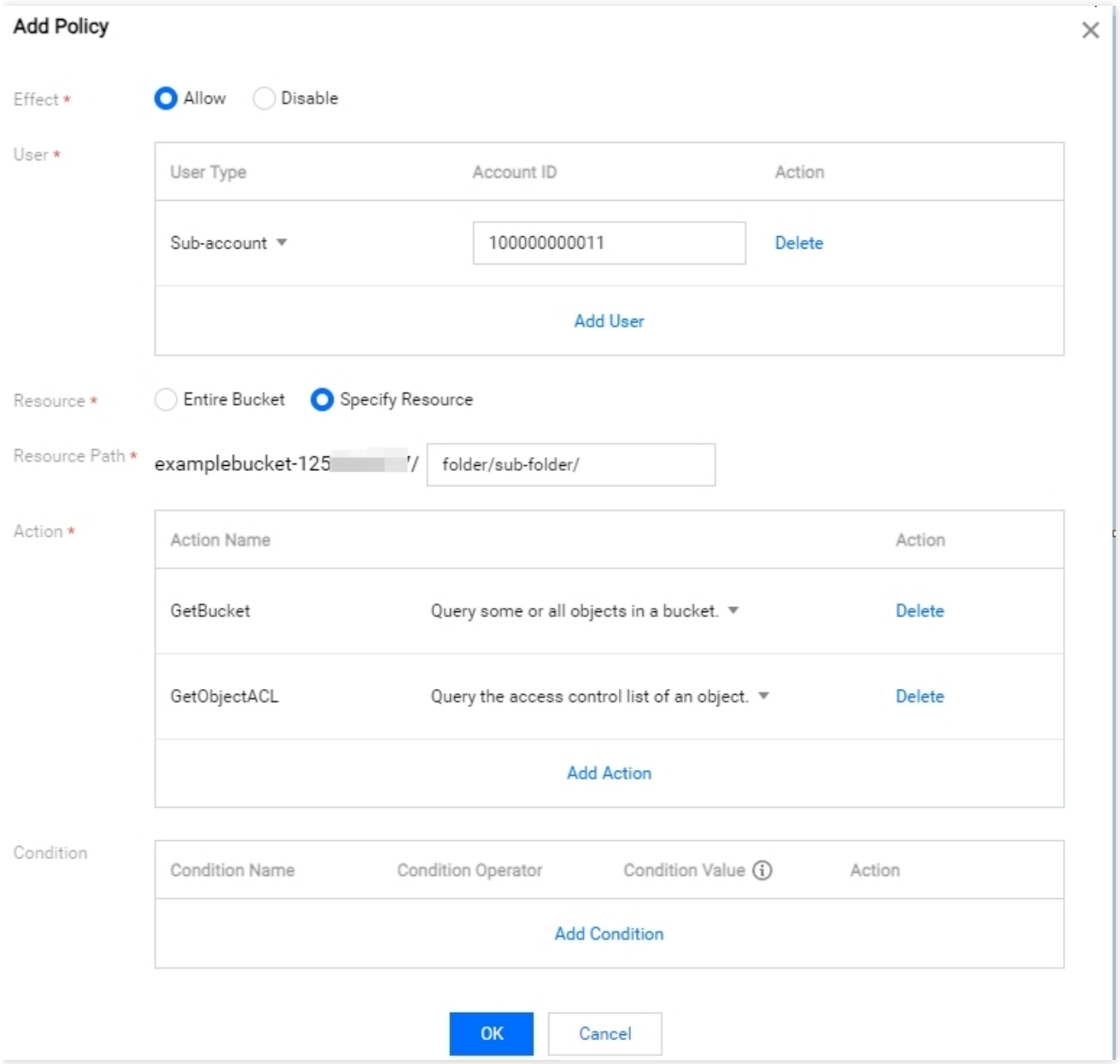


Case 2: Authorizing a Sub-account the Read Permission to Files in a Specific Directory

The following table describes the configuration information.

Configuration Item	Configuration Value

Configuration Item	Configuration Value
Effect	Allow
User Type	Sub-account
Account ID.	UIN of a sub-account, which should be a sub-account under the root account, for example, 100000000011
Resources	Specified resource
Resource path	Specified directory prefix, such as folder/sub-folder/ , which should be ended with /
Operation name	Read operation (including displaying the object list)

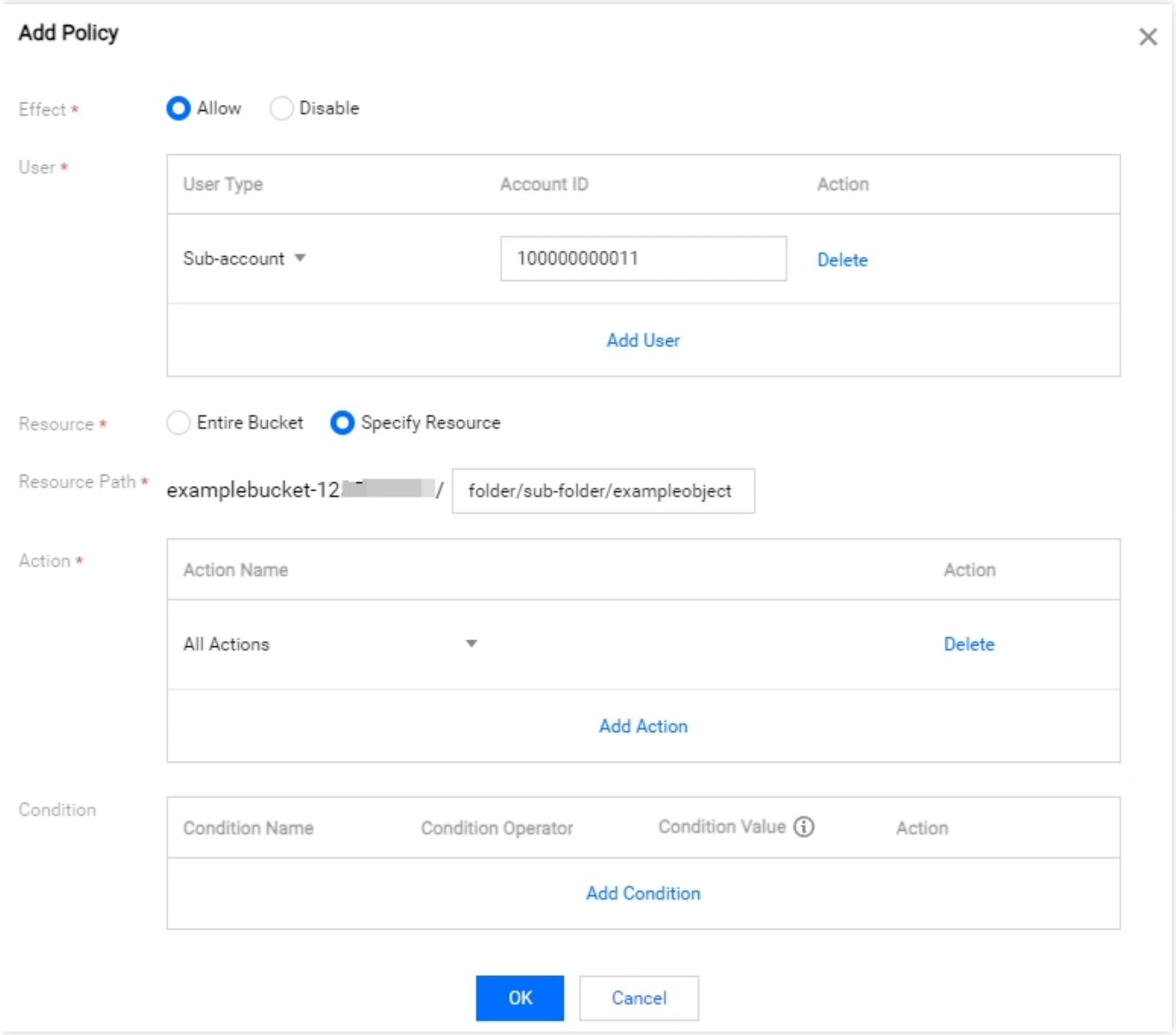


Case 3: Authorizing a Sub-account the Read and Write Permissions to a Specific File

The following table describes the configuration information.

Configuration Item	Configuration Value
Effect	Allow
User Type	Sub-account
Account ID.	UIN of a sub-account, which should be a sub-account under the root account, for example, 100000000011
Resources	Specified resource

Configuration Item	Configuration Value
Resource path	Specific object key, such as folder/sub-folder/exampleobject
Operation name	All operations



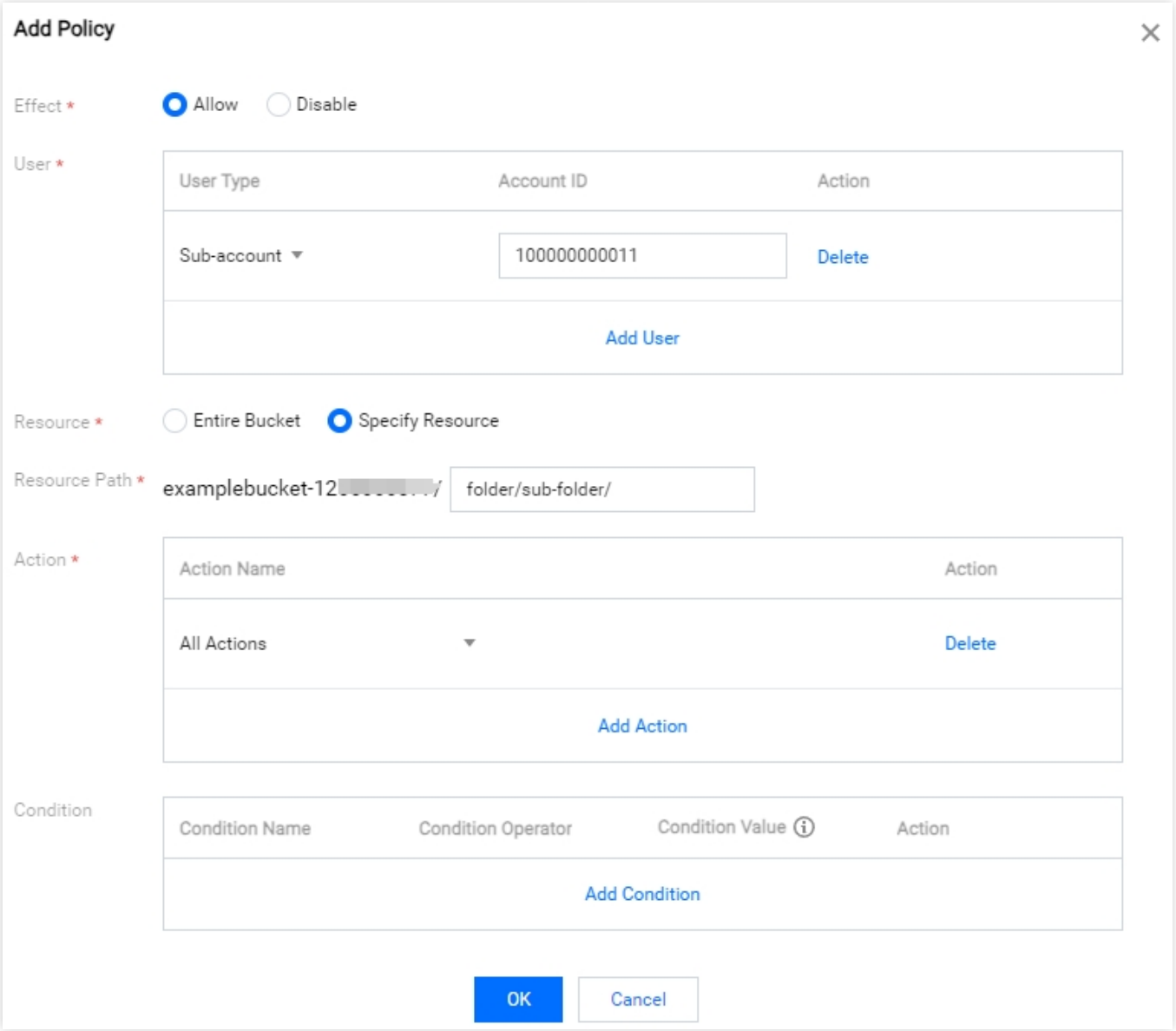
Case 4: Authorizing a Sub-account the Read and Write Permissions to All Files in a Specific Directory and Prohibiting the Read and Write Permissions to Specified Files in the Directory

In this case, two policies: Allow and Deny need to be added.

1. First, add the Allow policy. The following table describes the configuration information.

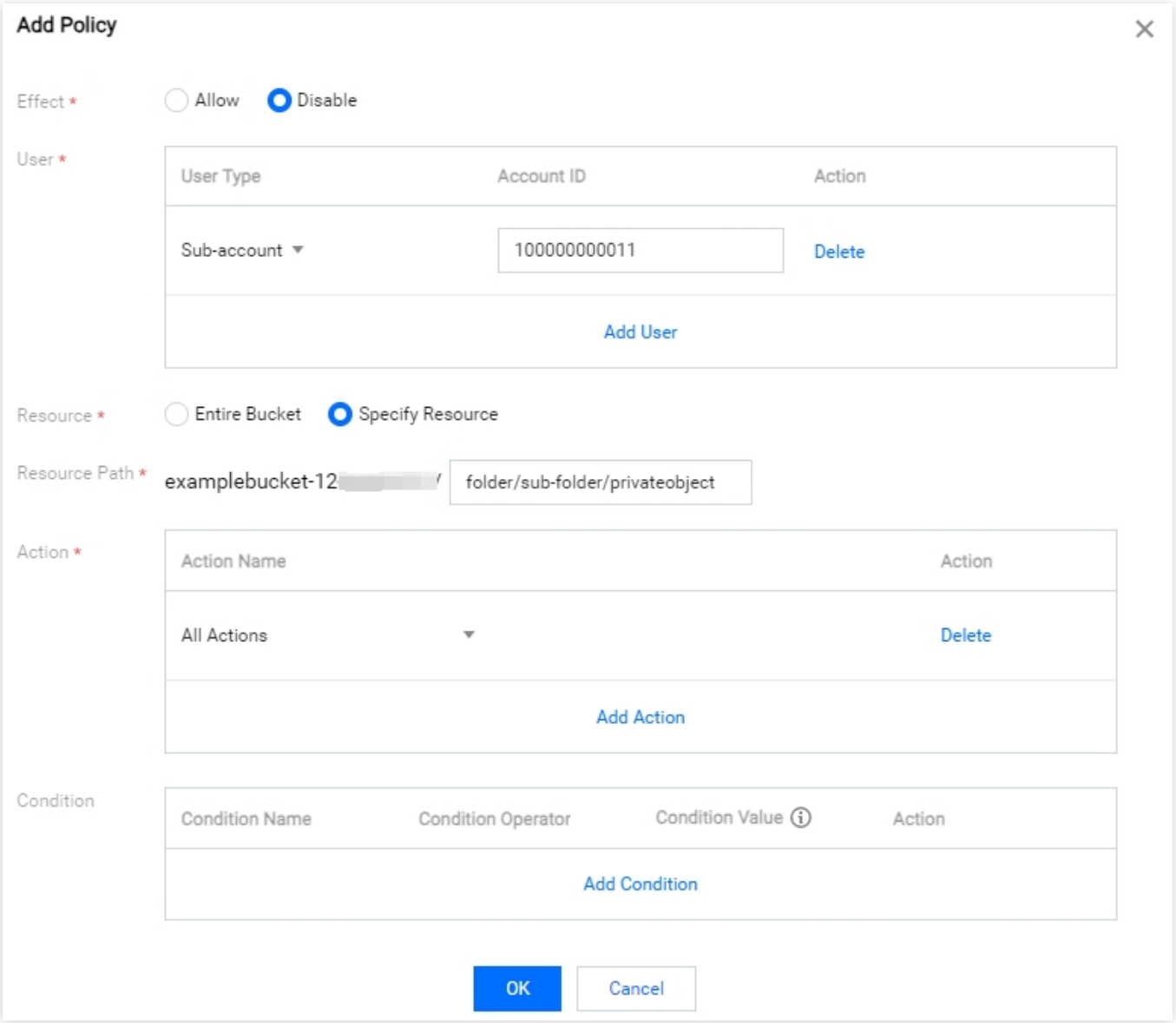
Configuration Item	Configuration Value

Configuration Item	Configuration Value
Effect	Allow
User Type	Sub-account
Account ID.	UIN of a sub-account, which should be a sub-account under the root account, for example, 100000000011
Resources	Specified resource
Resource path	Specified directory prefix, such as folder/sub-folder/ , which should be ended with /
Operation name	All operations



2. Then, add the Prohibit policy. The following table describes the configuration information.

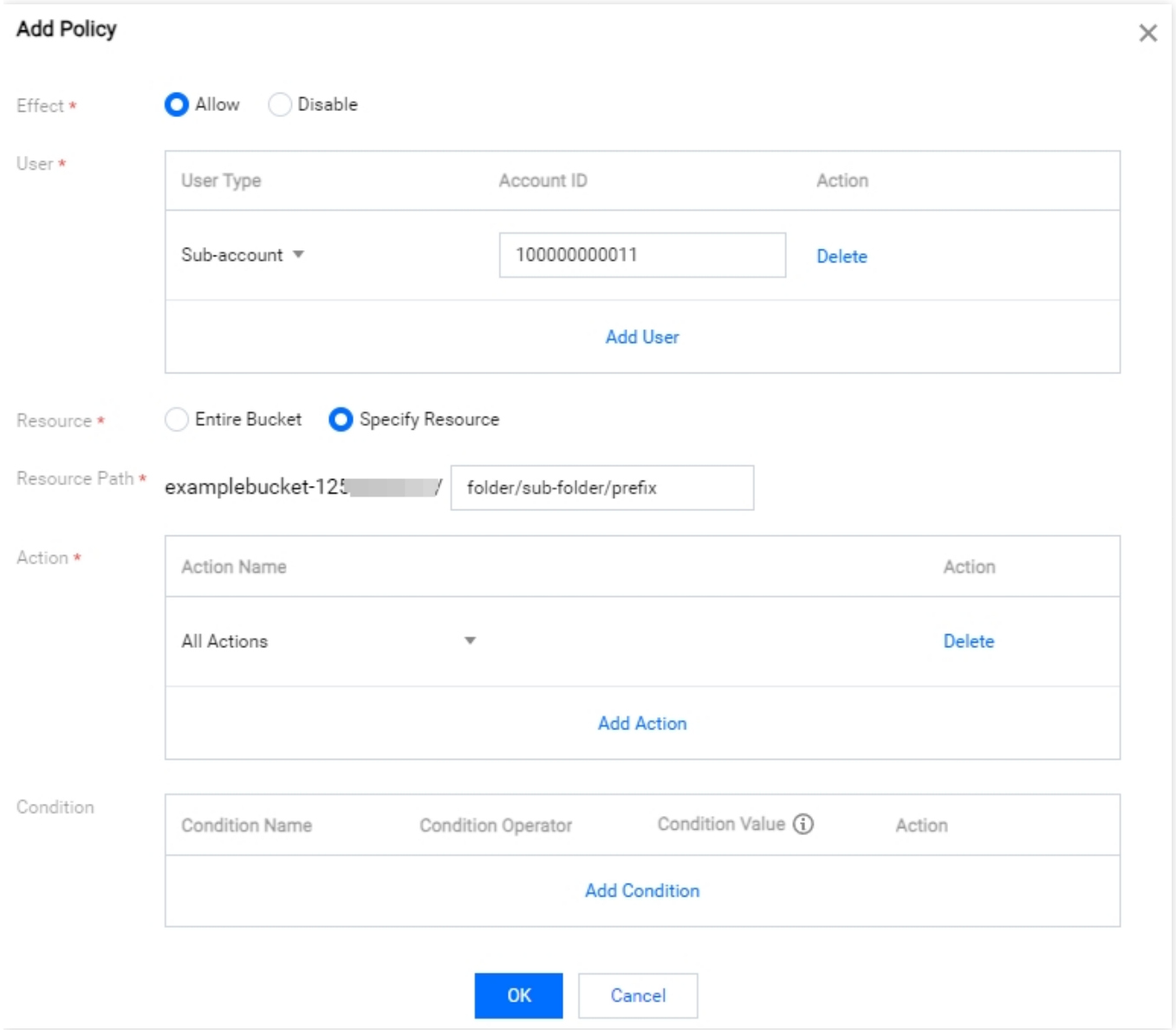
Configuration Item	Configuration Value
Effect	Prohibit
User Type	Sub-account
Account ID	UIN of a sub-account, which should be a sub-account under the root account, for example, 100000000011
Resources	Specified resource
Resource path	Object key to be prohibited to access, such as folder/sub-folder/privateobject
Operation name	All operations



Case 5: Authorizing a Sub-account the Read and Write Permissions to Files with a Specified Prefix

The following table describes the configuration information.

Configuration Item	Configuration Value
Effect	Allow
User Type	Sub-account
Account ID	UIN of a sub-account, which should be a sub-account under the root account, for example, 100000000011
Resources	Specified resource
Resource path	Specified prefix, such as folder/sub-folder/prefix
Operation name	All operations


Add Policy
Effect *
Allow
Disable
User *
User Type
Account ID
Action
Sub-account
100000000011
Delete
Add User
Resource *
Entire Bucket
Specify Resource
Resource Path *
examplebucket-125
/
folder/sub-folder/prefix
Action *
Action Name
Action
All Actions
Delete
Add Action
Condition
Condition Name
Condition Operator
Condition Value
Action
Add Condition
OK
Cancel

User Guide to COS API Authorization Policies

> Note:
>
> Grant the minimum API operation permissions to a sub-user or collaborator based on business requirements. If you grant the permissions of all resources（resource:*）or all operations（action:*）to a sub-user or collaborator, data security risks exist due to an excessive permission scope.

Overview

If a temporary key is used to access COS, different operation permissions are required for different COS API operations. In addition, you can specify the operation permissions for one operation or a series of operations at the same time.

The COS API authorization policy is a JSON string. For example, the following policy grants the permission to upload (including simple upload, form upload, and multipart upload) objects prefixed doc and download objects prefixed doc2 in the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
    "version": "2.0",
    "statement": [{
            "action": [
                //Simple upload operation
                "name/cos:PutObject",
                //Object upload through a form
                "name/cos:PostObject",
                //Multipart upload: Initializing multipart upload operations
                "name/cos:InitiateMultipartUpload",
                //Multipart upload: Listing ongoing multipart upload operations
                "name/cos:ListMultipartUploads",
                //Multipart upload: Listing uploaded parts
                "name/cos:ListParts",
                //Multipart upload: Uploading parts
                "name/cos:UploadPart",
                //Multipart upload: Completing all multipart upload operations
                "name/cos:CompleteMultipartUpload",
                //Canceling multipart upload operations
                "name/cos:AbortMultipartUpload"
            ],
            "effect": "allow",
```

```
            "resource": [
                "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc/*"
            ]
        },
        {
            "action": [
                //Downloading operation
                "name/cos:GetObject"
            ],
            "effect": "allow",
            "resource": [
                "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc2/*"
            ]
        }
    ]
}
```

Authorization Policy Element Description

Name	Description
version	Syntax version of a policy, which is 2.0 by default.
effect	The options include allow (permit) and deny (explicit refuse).
resource	Authorized data to be operated, which can be any resources, resources with a specified path prefix, resources with a specified absolute path, or resources with combinations of the above conditions.
action	COS API operation. You can specify an operation, the combination of a series of operations, or all operations (*) as required.
condition	Constraint conditions, which can be left blank. For details, see condition description.

The following describes examples for setting authorization policies of different COS APIs.

Service API

Querying the Bucket List

To grant access to the GET Service API, the action element of the policy is name/cos:GetService, and the resource element of the policy is * .

Example

The following policy grants the permission to query the bucket list.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:GetService"
      ],
      "effect": "allow",
      "resource": [
        "*"
      ]
    }
  ]
}
```

Bucket API

The resource element of the bucket API policy varies depending on the scenario:

- To operate buckets in all regions, the resource element of the policy is * .
- To operate buckets in a specified region, for example, to operate the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000, the resource element of the policy is qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/* .
- To operate a bucket with the specified name in the specified region, for example, to operate the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000, the resource element of the policy is qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/ .
  The value of the action element in a bucket API policy varies depending on the operation. The following describes all bucket API authorization policies.

Creating a Bucket

To grant access to the PUT Bucket API, set the action element of the policy to name/cos:PutBucket.

Example

The following policy grants the permission to create a bucket with any name whose region is ap-city1 and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
```

```
        "name/cos:PutBucket"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/"
      ]
    }
  ]
}
```

Retrieving Buckets and Their Permissions

To grant access to the HEAD Bucket API, set the action element of the policy to name/cos:HeadBucket.

Example

The following policy grants the permission to retrieve the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:HeadBucket"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/"
      ]
    }
  ]
}
```

Querying the Object List

To grant access to the GET Bucket API, set the action element of the policy to name/cos:GetBucket.

Example

The following policy grants the permission to query the list of objects in the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
```

```
      "action": [
        "name/cos:GetBucket"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/"
      ]
    }
  ]
}
```

Deleting a Bucket

To grant access to the Delete Bucket API, set the action element of the policy to name/cos:DeleteBucket.

Example

The following policy grants the permission to delete the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:DeleteBucket"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/"
      ]
    }
  ]
}
```

Setting a Bucket ACL

To grant access to the Put Bucket ACL API, set the action element of the policy to name/cos:PutBucketACL.

Example

The following policy grants the permission to set an ACL for the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
```

```
    {
      "action": [
        "name/cos:PutBucketACL"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/"
      ]
    }
  ]
}
```

Querying the ACL of a Bucket

To grant access to the GET Bucket acl API, set the action element of the policy to name/cos:GetBucketACL.

Example

The following policy grants the permission to obtain the ACL of the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:GetBucketACL"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/"
      ]
    }
  ]
}
```

Setting CORS Configuration

To grant access to the PUT Bucket cors API, set the action element of the policy to name/cos:PutBucketCORS.

Example

The following policy grants the permission to set the CORS configuration for the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:PutBucketCORS"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/"
      ]
    }
  ]
}
```

Querying CORS Configuration

To grant access to the GET Bucket cors API, the action element of the policy is name/cos:GetBucketCORS.

Example

The following policy grants the permission to query the CORS configuration of the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:GetBucketCORS"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/"
      ]
    }
  ]
}
```

Deleting CORS Configuration

To grant access to the DELETE Bucket cors API, set the action element of the policy to name/cos:DeleteBucketCORS.

Example

The following policy grants the permission to delete the CORS configuration for the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:DeleteBucketCORS"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/"
      ]
    }
  ]
}
```

Setting the Bucket Lifecycle

To grant access to the PUT Bucket lifecycle API, set the action element of the policy to name/cos:PutBucketLifecycle.

Example

The following policy grants the permission to set the lifecycle for the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:PutBucketLifecycle"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/"
      ]
    }
  ]
}
```

Querying the Bucket Lifecycle Configuration

To grant access to the GET Bucket lifecycle API, set the action element of the policy to name/

cos:GetBucketLifecycle.

Example

The following policy grants the permission to query the lifecycle configuration of the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:GetBucketLifecycle"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/"
      ]
    }
  ]
}
```

Deleting the Bucket Lifecycle Configuration

To grant access to the DELETE Bucket lifecycle API, set the action element of the policy to name/cos:DeleteBucketLifecycle.

Example

The following policy grants the permission to delete the lifecycle configuration of the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:DeleteBucketLifecycle"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/"
      ]
    }
  ]
}
```

Querying Multipart Upload

This API is used to query information about ongoing multipart uploads in a bucket. To grant access to this API, set the action element of the policy to name/cos:ListMultipartUploads.

Example

The following policy grants the permission to query ongoing multipart uploads in the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:ListMultipartUploads"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/"
      ]
    }
  ]
}
```

Object API

The resource element of an object API policy varies depending on the scenario:

- To operate all objects, the resource element of the policy is * .
- To operate objects in a specified bucket, for example, to operate objects in the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000, the resource element of the policy is qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/* .
- To operate objects in a specified bucket and with the specified path prefix, for example, to operate objects in the bucket whose name is examplebucket-1250000000, region is ap-city1, app ID is 1250000000, and path prefix is doc, the resource element of the policy is qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc/* .
- To operate objects in a specified absolute path, for example, to operate objects in the bucket whose name is examplebucket-1250000000, region is ap-city1, app ID is 1250000000, and absolute path is doc/audio.mp3 , the resource element of the policy is qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc/audio.mp3 .

  The value of the action element in an object API policy varies depending on the operation. The

following describes all object API authorization policies.

Uploading Objects in Simple Upload Mode

To grant access to the PUT Object API, set the action element of the policy to name/cos:PutObject.

Example

The following policy grants the permission to upload objects prefixed doc in simple upload mode to the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
 "version": "2.0",
 "statement": [
  {
   "action": [
    "name/cos:PutObject"
   ],
   "effect": "allow",
   "resource": [
    "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc/*"
   ]
  }
 ]
}
```

Multipart Upload

Multipart upload operations include Initiate Multipart Upload, List Multipart Uploads, List Parts, Upload Part, Complete Multipart Upload, and Abort Multipart Upload. To grant access to this API, set the action element of the policy to a set of "name/cos:InitiateMultipartUpload","name/cos:ListMultipartUploads","name/cos:ListParts","name/cos:UploadPart","name/cos:CompleteMultipartUpload","name/cos:AbortMultipartUpload" .

Example

The following policy grants the permission to upload objects prefixed doc in multipart upload mode to the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
 "version": "2.0",
 "statement": [
  {
   "action": [
    "name/cos:InitiateMultipartUpload",
    "name/cos:ListMultipartUploads",
```

```
        "name/cos:ListParts",
        "name/cos:UploadPart",
        "name/cos:CompleteMultipartUpload",
        "name/cos:AbortMultipartUpload"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc/*"
      ]
    }
  ]
}
```

Uploading Objects Through Forms

To grant access to the POST Object API, set the action element of the policy to name/cos:PostObject.

Example

The following policy grants the permission to upload objects prefixed doc in POST upload mode to the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:PostObject"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc/*"
      ]
    }
  ]
}
```

Querying Metadata of Objects

To grant access to the HEAD Object API, set the action element of the policy to name/cos:HeadObject.

Example

The following policy grants the permission to query objects prefixed doc in the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
```

```
  "version": "2.0",
  "statement": [
   {
     "action": [
       "name/cos:HeadObject"
     ],
     "effect": "allow",
     "resource": [
       "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc/*"
     ]
   }
  ]
}
```

Downloading an Object

To grant access to the GET Object API, set the action element of the policy to name/cos:GetObject.

Example

The following policy grants the permission to download objects prefixed doc in the bucket whose name is examplebucket-1250000000, region is ap-city1 and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
   {
     "action": [
       "name/cos:GetObject"
     ],
     "effect": "allow",
     "resource": [
       "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc/*"
     ]
   }
  ]
}
```

Copying Objects

To grant access to the Put Object Copy API, set the action element of the policy for the destination object to name/cos:PutObject and the action element of the policy for the source object to name/cos:GetObject.

Example

The following policy grants the permission to use multipart copy to copy objects from the path prefixed with doc to the path prefixed with doc2 in the bucket whose name is examplebucket-1250000000, region is

ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:PutObject"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc/*"
      ]
    },
    {
      "action": [
        "name/cos:GetObject"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc2/*"
      ]
    }
  ]
}
```

"qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc2/*" is the source object.

Copying Parts

To grant access to the Upload Part - Copy API, set the action element of the policy for the target object to a set of "name/cos:InitiateMultipartUpload","name/cos:ListMultipartUploads","name/cos:ListParts","name/cos:PutObject","name/cos:CompleteMultipartUpload","name/cos:AbortMultipartUpload" , and the action element of the policy for the source object to name/cos:GetObject.

Example

The following policy grants the permission to use multipart copy to copy objects from the path prefixed with doc to the path prefixed with doc2 in the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
```

```
        "name/cos:InitiateMultipartUpload",
        "name/cos:ListMultipartUploads",
        "name/cos:ListParts",
        "name/cos:PutObject",
        "name/cos:CompleteMultipartUpload",
        "name/cos:AbortMultipartUpload"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc/*"
      ]
    },
    {
      "action": [
        "name/cos:GetObject"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc2/*"
      ]
    }
  ]
}
```

"qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc2/*" is the source object.

Setting an ACL for Objects

To grant access to the Put Object ACL API, set the action element of the policy to name/cos:PutObjectACL.

Example

The following policy grants the permission to set an ACL for objects prefixed doc in the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:PutObjectACL"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc/*"
      ]
    }
```

```
  ]
}
```

Querying an Object ACL

To grant access to the Get Object ACL API, the action element of the policy is name/cos:GetObjectACL.

Example

The following policy grants the permission to query the ACL of objects prefixed doc in the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:GetObjectACL"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc/*"
      ]
    }
  ]
}
```

Checking CORS Configuration

To grant access to the OPTIONS Object API, the action element of the policy is name/cos:OptionsObject.

Example

The following policy grants the permission to send Options requests under the bucket whose name is examplebucket-1250000000, region is ap-city1, app ID is 1250000000, and path prefix is doc.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:OptionsObject"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc/*"
      ]
```

```
    }
  ]
}
```

Recovering Archived Objects

To grant access to the Post Object Restore API, the action element of the policy is name/cos:PostObjectRestore.

Example

The following policy grants the permission to recover archived objects prefixed doc in the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:PostObjectRestore"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc/*"
      ]
    }
  ]
}
```

Deleting an Object

To grant access to the DELETE Object API, the action element of the policy is name/cos:DeleteObject.

Example

The following policy grants the permission to delete the object named audio.mp3 in the bucket whose name is examplebucket-1250000000, region is ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:DeleteObject"
      ],
      "effect": "allow",
```

```
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/audio.mp3"
      ]
    }
  ]
}
```

Deleting Multiple Objects

To grant access to the DELETE Multiple Objects API, the action element of the policy is name/cos:DeleteObject .

Example

The following policy grants the permission to batch delete objects named audio.mp3 and video.mp4 in the bucket whose name is examplebucket-1250000000, region ap-city1, and app ID is 1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:DeleteObject"
      ],
      "effect": "allow",
      "resource": [
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/audio.mp3",
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/video.mp4"
      ]
    }
  ]
}
```

Authorization Policies in Common Scenarios

Granting Full Read and Write Permissions to All Resources

The following policy grants full read and write permissions to all resources.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
```

```
        "*"
      ],
      "effect": "allow",
      "resource": [
        "*"
      ]
    }
  ]
}
```

Granting the Read-Only Permission to All Resources

The following policy grants the read-only permission to all resources.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "name/cos:HeadObject",
        "name/cos:GetObject",
        "name/cos:ListObject",
        "name/cos:OptionsObject"
      ],
      "effect": "allow",
      "resource": [
        "*"
      ]
    }
  ]
}
```

Granting Read and Write Permissions to Resources with a Specified Path Prefix

The following policy grants the access permission only for files with the path prefix doc in the bucket named examplebucket-1250000000.

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "*"
      ],
      "effect": "allow",
      "resource": [
```

```
        "qcs::cos:ap-city1:uid/1250000000:examplebucket-1250000000/doc/*"
      ]
    }
  ]
}
```

Security Guidelines for Temporary Keys Used for Direct Transmission from the Frontend to COS

Overview

In mobile and web applications, you can initiate requests to COS directly on the frontend by using the iOS/Android/JavaScript SDK. Data upload and download do not pass through the backend server. In this way, you can reduce the bandwidth usage and load of the backend server and make full use of the COS bandwidth, improving the user experience of your application.

In actual application, you need to use a temporary key as the signature for frontend COS requests to avoid problems, such as leakage of the permanent key and unauthorized access. However, if you specify excessive permissions or paths when generating a temporary key, even though a temporary key is used, problems such as unauthorized access may still occur, which brings risks to your application. This document describes some risk cases and security regulations that you need to comply with for your application to securely use COS.

Prerequisites

This document assumes that you fully understand the concepts related to temporary keys and can generate and use temporary keys to send requests to COS.

Risk Cases and Security Regulations

Risk Case 1: The Value Range of the resource Field Exceeds the Limit

Application A uses COS when registered users upload avatars. Each registered user's avatar has a fixed object key app/avatar/<Username>.jpg , and different avatar sizes are included, corresponding to object keys app/avatar/<Username>_m.jpg and app/avatar/<Username>_s.jpg , respectively. For convenience, the backend directly sets the resource field to <BucketName-APPID>/app/avatar/* when generating a temporary key. Malicious users can obtain the generated temporary key through network packet capturing and other means and overwrite any user's avatar, leading to unauthorized access and loss of legitimate avatar data.

Security Regulations

The resource field indicates the resource path that can be accessed with a temporary key, and the users covered by this path need to be fully considered. In principle, resources specified by the resource field should only be used by a single user. The specified `<BucketName-APPID>/app/avatar/*` in this case obviously covers all users. Therefore, a security vulnerability exists.

In this case, you can change the user's avatar path to `app/avatar/<Username><size>.jpg` . You can set the resource field to `<BucketName-APPID>/app/avatar/<Username>/*` to meet the regulation requirements. In addition, multiple values of the resource field can be sent in the form of an array. Therefore, you can also explicitly specify multiple resource values to limit the final resource paths the user is authorized to access, for example:

```
"resource": [
    "<BucketName-APPID>/app/avatar/<Username>.jpg",
    "<BucketName-APPID>/app/avatar/<Username>_m.jpg",
    "<BucketName-APPID>/app/avatar/<Username>_s.jpg"
]
```

Risk Case 2: The Value Range of the action Field Exceeds the Limit

Application B offers a public photo wall feature, and all photos are stored under `app/photos/*` . The client needs to list objects (GET Bucket) and download objects (GET Object). For convenience, the backend directly sets the action field to `name/cos:*` when generating a temporary key. After malicious users obtain the generated temporary key through means like network packet capturing, they can perform all object operations (such as upload and delete) on any object under this resource path, leading to unauthorized access, which causes data loss and affects online businesses.

Security Regulations

The action field indicates the operation requests that are permitted by a temporary key. In principle, temporary keys that allow all operations, such as `name/cos:*` should not be issued to the frontend. All necessary operations should be explicitly listed, and if the resource paths required for different operations vary, the action and resource paths should be matched separately, rather than specifying them in batches.

In this case, you should use `"action": [ "name/cos:GetBucket", "name/cos:GetObject" ]` to specify specific operations.

Risk Case 3: The Value Ranges of the action and resource Fields Exceed the Limit

Application C offers a management tool that allows users to list and download everyone's files (`app/files/*`) but can only upload and delete files in their own directories (`app/files/<Username>/*`). For convenience, the backend mixes the two permissions, that is, four operations (actions) together when generating a temporary key. The resource paths corresponding to the two permissions are also mixed. As a result, the temporary key has more extensive permissions specified in the resource paths, that is, users can list, download, upload, and delete anyone's files. In this way, malicious users can alter or delete others' files,

leading to unauthorized access and legitimate data disclosure.

Security Regulations

Multiple actions and resources should not be simply combined separately and should be combined by using multiple statements to avoid more excessive permissions due to simple separated combinations.

In this case, the following statements should be used:

```
"statement": [
    {
        "effect": "allow",
        "action": [
            "name/cos:GetBucket",
            "name/cos:GetObject"
        ],
        "resource": "<BucketName-APPID>/app/files/*"
    },
    {
        "effect": "allow",
        "action": [
            "name/cos:PutObject",
            "name/cos:DeleteObject"
        ],
        "resource": "<BucketName-APPID>/app/files/<Username>/*"
    }
]
```

Risk Case 4: Unauthorized Access to a Temporary Key

Application D provides a forum application, and posts and attachments in the forum are stored in COS. Users of the forum application are classified into different levels, and only active users with a certain number of posts can view posts and attachments in certain subforums. For some public subforums, all users can view the posts and attachments.
When COS is used, the API used to generate a temporary key at the backend generates a temporary key for downloading attachments in a subforum based on the subforum ID transmitted from the frontend. However, the backend does not determine whether a user who sends a request has permission to access the subforum specified by the subforum ID. As a result, anyone can request to obtain a temporary key for obtaining attachments in a private subforum over this API, resulting in unauthorized access, and resources that require limited access are not effectively limited, resulting in data leakage.

Security Regulations

In addition to ensuring that the permissions of the obtained temporary key are granted as expected, the API for obtaining a temporary key also needs to check whether the correct permissions are requested by

correct users based on the actual business scenario. This prevents users with low permissions from obtaining temporary keys for high permissions.

In this case, the backend API should also check whether the requesting user has permission to access the specified subforum. If not, it should not return a temporary key.

Summary

The above cases illustrate the security risks that may occur if permissions of a temporary key are more excessive than expected. In scenarios where data can be directly uploaded to COS on the frontend, malicious users can obtain a temporary key more easily than in the scenario where COS is accessed on the backend. Therefore, developers should pay more attention to permission control.

The security regulation, that is, the minimum permission principle mentioned in this document, can enumerate all possible permissions based on the action and resource fields in actual applications. For example, in the case of three actions and two resources, six (2 x 3) accessible resources and the corresponding actions can be calculated. You can determine whether each scenario is within the expected permission scope. If not, enumerate multiple statements to split permissions.

In addition, the API used to generate a temporary key should also fully consider identity authentication and authorization processing. Only if the behavior of obtaining a temporary key is secure, the obtained temporary key is truly secure. No oversight is allowed in the security chain.

Authorizing a Sub-account Under Another Root Account to Operate Buckets Under the Local Root Account

Overview

Two buckets examplebucket1-1250000000 and examplebucket2-1250000000 exist under root account A (whose app ID is 1250000000). Due to business requirements, sub-account B0 under root account B wants to operate the two buckets under root account A. The following describes how to authorize the permission.

Directions

Authorizing Root Account B to Operate Buckets Under Root Account A

1. Use the root account A to log in to the COS console.
2. Click Bucket List, locate the bucket to be authorized, and click the bucket name to enter the bucket details page.
3. Click the Permission Management tab to switch to the Permission Management page.
4. Locate Policy Permission Settings, click Add Policy, and enter the following form items:
    - Effect: Allowed.
    - User: Click Add User. Select Root Account for User Type and enter the UIN of root account B, such as 100000000002 for Account ID.
    - Resource: Select resources as required. By default, the entire bucket is selected.
    - Resource Path: It needs to be specified only when Resource is specified. Specify it as required.
    - Action: Click Add Action. Select all operations. If only some operations need to be authorized, you can select one or more required operations.
    - Condition: Enter it as required. If it is not required, leave it blank.
5. Click OK. In this case, root account B is granted relevant permissions to operate the bucket.
6. To authorize relevant permissions to operate other buckets, repeat the preceding steps.

Authorizing Sub-Account B0 to Operate Buckets Under Account A

1. Use root account B to log in to the CAM console and go to the Policy Management page.

2. Select Create Custom Policy > Create by Policy Syntax, select a blank template, and click next step: Modify Policy.

3. Enter the following form:

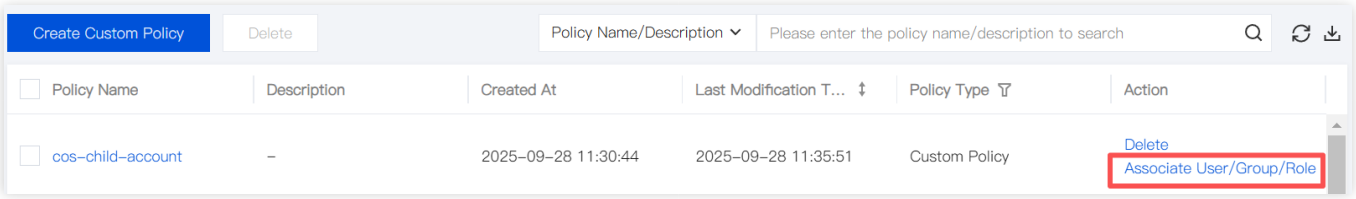
- Policy Name: Define a unique and meaningful policy name, such as cos-child-account.
- Remarks: Enter it at random, which is optional.
- Modify Policy Content:

```
{
    "version": "2.0",
    "statement": [
        {
            "action": "cos:*",
            "effect": "allow",
            "resource": "qcs::cos::uid/1250000000:examplebucket1-1250000000/*"
        }
    ]
}
```

1250000000 in uid/1250000000 indicates the UID of root account A, and examplebucket1-1250000000 is the name of the authorized bucket. The bucket resource examplebucket1-1250000000/* can also be *, indicating that all buckets under account A that root account B has permission to operate are authorized to sub-account B0.

4. Click Finish, and then the policy is created.

5. Find the created policy in the policy list and click Associate User/Group/Role on the right.



6. In the pop-up window, select sub-account B0 and click OK.
   After authorization, you can use the key of sub-account B0 to operate buckets under account A.

Performance Optimization
Request Rate and Performance Optimization

This section describes the best practices for request rate optimization in COS.

COS supports a typical workload capacity of 100 PUT/LIST/DELETE requests per second or 400 GET requests per second. If your workload exceeds the threshold, you can follow this guide to optimize your request rate.

> Note:
>
> The request load indicates the number of requests initiated per second rather than the number of concurrent connections. In other words, you can send hundreds of new connection requests per second while maintaining thousands of existing connections.

COS supports performance scaling to support higher request rates. If the request rate of a bucket is expected to exceed 500 PUT/LIST/DELETE requests per second, you are advised to contact us to prepare for the workload and avoid request limits.

> Note:
>
> If you have a mixed request load that only occasionally reaches 500 requests per second and does not exceed 800 requests per second during bursts, you can ignore this guide.

Mixed Request Load

When a large number of objects need to be uploaded, the object key you select may cause performance issues. The following simply describes how COS stores object key-values.

In each service region of COS, the cloud platform maintains key-values of buckets and objects as indexes. The object keys are stored in UTF-8 binary order across multiple index partitions. For a large number of key-values, using timestamps or alphabetical order may exhaust the read/write performance of the partition where the key-values are stored. The bucket path testbucket-123454321.cos.city.yfm4.fsphere.cn is used as an example. The following lists some cases that may exhaust index performance.

20170701/log120000.tar.gz
20170701/log120500.tar.gz
20170701/log121000.tar.gz
20170701/log121500.tar.gz

```
...
image001/indexpage1.jpg
image002/indexpage2.jpg
image003/indexpage3.jpg
...
```

If the typical load of your business exceeds 500 requests per second, you should avoid using sequential key-values as mentioned in the above case. When your business should use sequential numbers or date-time characters as object keys, you can add random prefixes to the object key names. This ensures key-value management across multiple index partitions and improves the performance under concentrated loads. The following describes some methods to increase randomness in key-values.

All of the following methods can enhance the access performance of a single bucket. If the typical load of your business exceeds 500 requests per second, in addition to the following methods, you still need to contact us to make preparations for your business load in advance.

Adding Hexadecimal Hash Prefixes

The most direct way to increase the randomness of object keys is to add a hash string to the beginning of an object key name as a prefix. For example, when an object is uploaded, you can perform SHA1 or MD5 hash calculation on the path key-value, pick several characters as a prefix, and add it to the key name. Usually, a hash prefix of 2 to 4 characters can meet the requirements.

```
faf1-20170701/log120000.tar.gz
e073-20170701/log120500.tar.gz
333c-20170701/log121000.tar.gz
2c32-20170701/log121500.tar.gz
```

> Note:
>
> The COS key-values are indexed in UTF-8 binary order. When performing the operation of listing objects (GET Bucket), you might need to initiate 65,536 such operations to obtain the original complete prefix structure of 20170701.

Adding Enumerated Value Prefixes

To ensure that your object keys are easily retrievable, you can enumerate some prefixes based on your file type to group your objects. Prefixes with the same enumeration value share the performance of the index partition where they are located.

```
logs/20170701/log120000.tar.gz
logs/20170701/log120500.tar.gz
logs/20170701/log121000.tar.gz
...
```

images/image001/indexpage1.jpg
images/image002/indexpage2.jpg
images/image003/indexpage3.jpg
...

If the access load under the same enumerated prefix remains high, exceeding 500 requests per second, you can refer to the method mentioned above to add a hexadecimal hash prefix after the enumerated value in multiple index partitions for higher read/write performance.

logs/faf1-20170701/log120000.tar.gzlogs/
e073-20170701/log120500.tar.gz
logs/333c-20170701/log121000.tar.gz
...
images/0165-image001/indexpage1.jpg
images/a349-image002/indexpage2.jpg
images/ac00-image003/indexpage3.jpg
...

Reversing the Key Name String

When your business has to use incremental IDs or dates or needs to upload a large number of objects with successive prefixes at a time, use the following typical method:

20170701/log0701A.tar.gz
20170701/log0701B.tar.gz
20170702/log0702A.tar.gz
20170702/log0702B.tar.gz
...
id16777216/album/city/img20170701121314.jpg
id16777216/music/artist/tony/anythinggoes.mp3
id16777217/video/record20170701121314.mov
id16777218/live/show/date/20170701121314.mp4
...

The above key-value naming method easily depletes the index partition where key-values with prefixes `2017` and `id` are located. In this case, reversing part of the key-value prefixes to ensure randomness.

10707102/log0701A.tar.gz
10707102/log0701B.tar.gz
20707102/log0702A.tar.gz
20707102/log0702B.tar.gz
...
61277761di/album/city/img20170701121314.jpg
61277761di/music/artist/tony/anythinggoes.mp3

71277761di/video/record20170701121314.mov
81277761di/live/show/date/20170701121314.mp4
...

Data Migration
Migrating Data Within COS

Practice Background

To migrate data from one bucket to another within COS, COS users can use the following migration methods:

In-cloud replication>In-cloud replication

In-cloud replication is a COS configuration item for buckets. By setting in-cloud replication rules, you can automatically and asynchronously copy incremental objects in buckets across different regions. Once in-cloud replication is enabled, COS precisely copies the object content in the source bucket (such as the object metadata and version ID) to the destination bucket, and the copied object replicas have completely consistent attributes with the source objects. In addition, operations on objects in the source bucket, such as uploading or deleting objects, are also copied to the destination bucket.
The destination bucket should also use COS for object storage.

Cross-cloud replication>Cross-cloud replication

Cross-cloud replication is an advanced feature provided by COS that enables object migration between different clouds through cross-cloud replication rule configuration. This feature supports copying objects to any COS destination bucket that supports the S3 protocol, allowing object replication and migration as long as the network is smooth.
Cross-cloud replication can precisely copy object content in the source bucket, including the object metadata and version ID. The copied object replicas have completely consistent attributes with the source objects. Compared to in-cloud replication, cross-cloud replication provides more powerful capabilities with broader coverage, suitable for various scenario requirements.

Data Disaster Recovery and Backup Disaster Recovery and High Availability Architecture Based on Replication Within the Cloud

Overview

COS provides customers with 99.95% availability and 99.999999999% reliability. However, due to many uncontrollable factors such as natural disasters and fiber optic failures, neither the availability nor the reliability of cloud data can be 100%. Meanwhile, for certain industries, such as the financial industry, high availability and high reliability need to be ensured for their business due to the special nature of their business.

In order to achieve business continuity and stability and meet the needs of enterprises for high availability and high reliability, COS provides a data disaster recovery and high-availability solution based on the inter-cloud replication feature. We recommend that when enterprises use the cloud, they perform disaster recovery backup of cloud data according to business needs to ensure continuous and stable business operations.

This document mainly introduces two aspects. First, it introduces a disaster recovery solution for the primary-secondary switch of the cloud workload based on in-cloud replication. Second, it further introduces a high-availability solution based on in-cloud replication, which achieves the high availability of the workload through in-cloud replication, origin-pull, SCF, and other products and features.

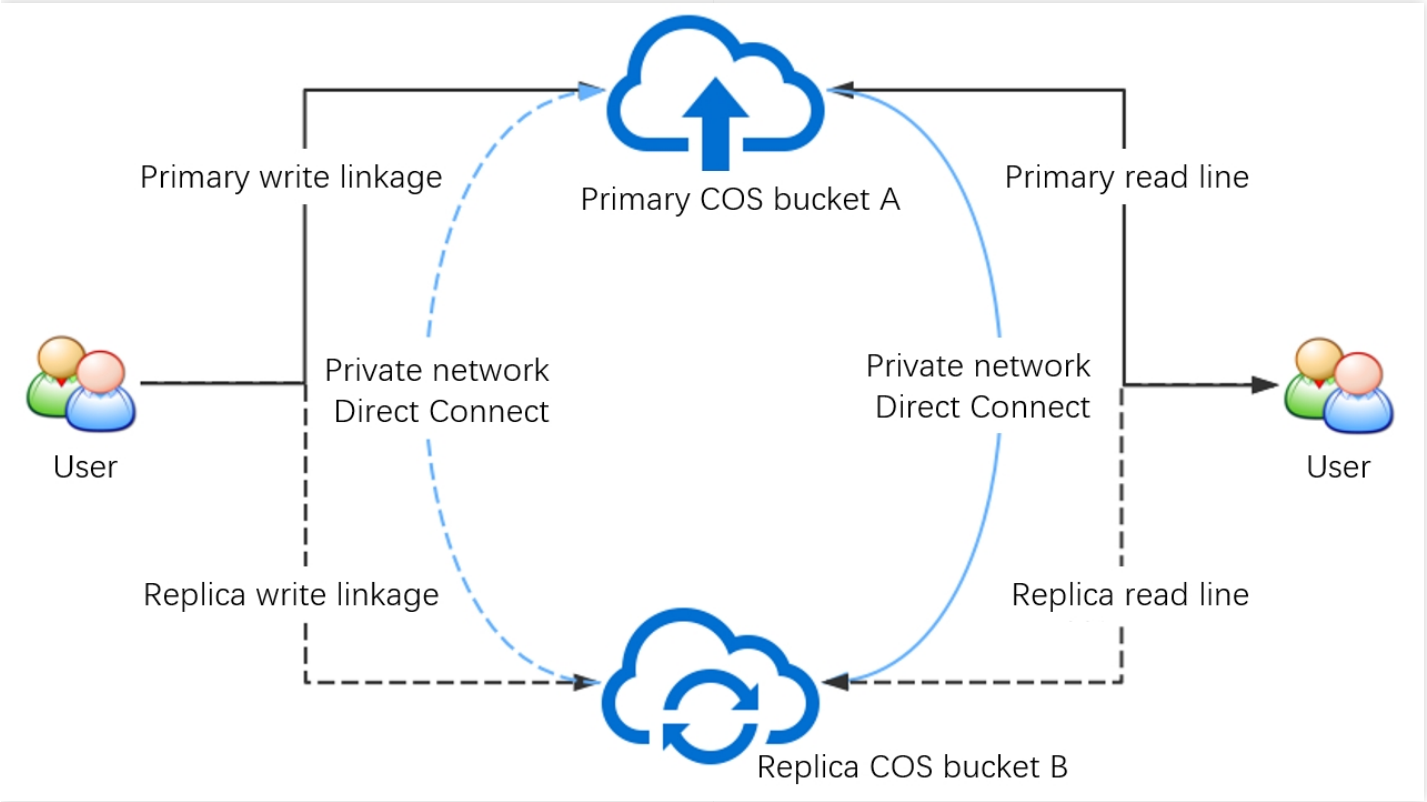
DR and Backup Solutions Based on Inter-Cloud Replication

Disaster recovery needs to meet three elements: Redundance, Remote, and Replication.

- Redundance: That is, data redundancy, which requires that data be backed up to another available system at the same time.
- Remote: Refers to the storage of backup data in another region that is far away. Because disasters often have geographical continuity, only a sufficiently long distance can guarantee the availability of redundant data.
- Replication: Refers to zero loss of backup data.
  COS's inter-cloud replication feature can achieve inter-cloud synchronization of incremental data. The data uploaded by users can be copied to a storage bucket in another region within a few seconds to

tens of minutes, depending on the file size and the distance between regions. Based on inter-cloud replication, data can be backed up in different locations for redundancy, thus achieving business disaster recovery. For an introduction to inter-cloud replication, see Inter-Cloud Replication Overview. To enable inter-cloud replication, you need to enable the versioning feature first. For an introduction to versioning, see Versioning Overview.

The following figure shows the architecture diagram of DR and backup based on inter-cloud replication.



In this architecture, the customer's storage bucket A and storage bucket B serve as the primary/replica of each other. In this architecture, the customer's storage bucket A and storage bucket B serve as the primary/replica of each other. To ensure business continuity and stability, the enterprise configured inter-cloud replication rules for bucket A and bucket B, respectively. When the inter-cloud replication rule is in effect, the incremental data in bucket A will be automatically copied to bucket B, and the incremental data in bucket B will also be automatically copied to bucket A.

> Note:
>
> After the incremental data in bucket A is copied to bucket B, although it is the incremental data in bucket B, it will not be copied to bucket A again.

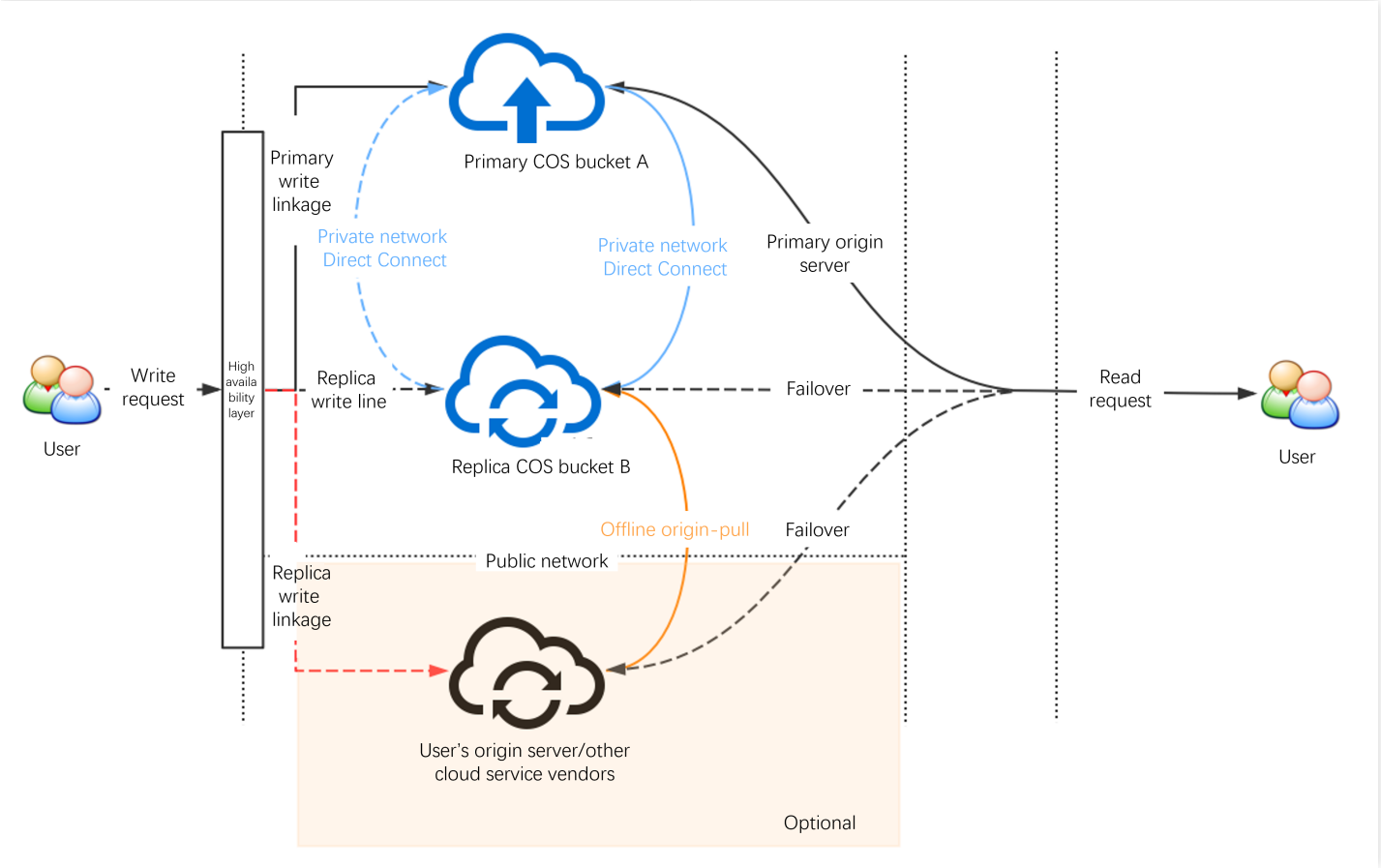
Under normal circumstances, the enterprise's primary read/write request links all points to bucket A, and all incremental data will be automatically and incrementally synchronously copied to bucket B as backup data. The client can add a network quality detection module to the upload or download program. When the

primary bucket A is detected to be down, the read and write request link can be quickly switched to the replica bucket B.

High-Availability Solution Based on Inter-Cloud Replication

The above section introduced a disaster recovery backup solution based on in-cloud replication, which utilizes existing cloud products and features to implement data backup and disaster recovery switching. However, the actual workload operation status may be complex and diverse, and the above-mentioned disaster recovery backup solution may not be able to guarantee the high availability of the workload. Therefore, this section proposes a high-availability solution based on in-cloud replication, which achieves the high availability of the workload through in-cloud replication, origin-pull, SCF, and other products and features.

The following figure shows the architecture diagram of business HA based on inter-cloud replication.



This architecture is mainly divided into the following levels:

- HA layer: Integrates network detection and business scheduling and switches links based on metrics, such as link connectivity. Users can implement it based on SCF (refer to the introduction in the previous section) or implement it on the client based on business requirements.

- Storage layer: Generally, it consists of buckets of COS in different regions.

The way this architecture ensures high availability of business is described as follows:

1. Under normal circumstances, the enterprise's primary write request link points to bucket A, and all incremental data will be automatically and synchronously copied to bucket B as backup data.
2. When the link to primary bucket A is not accessible (for example, the dial quality decreases or upload failure is detected), the client can switch the write request link to primary bucket B. At this time, all incremental data will also be automatically and synchronously copied to bucket A.
3. Customers can also choose to back up redundant data on their own origin server or other cloud vendors and configure an origin-pull policy for bucket B. Assume that in an extreme case, the links of primary buckets A and B are both unavailable at the same time. Then, if uploading data to bucket B fails, bucket B can pull data from the origin server.

> Note:
>
> - The cost of a full redundant backup of data is high. Customers can also choose to only redundantly back up hot data (such as files uploaded within a few hours) to reduce data storage costs.
> - If you choose the origin server as part of your high-availability architecture, be careful to assess the origin server bandwidth and the impact of its limitations when designing the architecture.

Reference Document

The following documents may help you implement a disaster recovery and highly-availability architecture:

- Versioning Overview
- Inter-Cloud Replication Overview

Domain Name Management Practice
Setting Cross-Origin Access

One-Origin Policy

The one-origin policy restricts the way that files/scripts loaded from one origin interact with resources from another origin and is a key security mechanism for isolating potential malicious files. Resources with the same protocol, domain name (or IP address), and port are considered to belong to the same origin. Scripts in one origin can only be used to read/write resources in the origin and cannot be used to access resources in other origins. This security limit is called a one-origin policy.

One-Origin Definition

Webpages that have the same protocol, domain name, and port (if specified) are regarded as from the same origin. The following table describes how to test whether a webpage belongs to the same origin as http://www.example.com/dir/page.html :

URL	Result	Possible Causes
http://www.example.com/dir2/other.html	Succeeded	Same protocol, domain name, and port
http://www.example.com/dir/inner/another.html	Succeeded	Same protocol, domain name, and port
http://www.example.com/secure.html	Failed	Different protocols (https)
http://www.example.com:81/dir/etc.html	Failed	Different ports (81)
http://news.example.com/dir/other.html	Failed	Different domain names

Cross-Origin Access

Cross-origin resource sharing (CORS) is also known as cross-origin access. It allows web application servers to control cross-origin access to ensure secure cross-origin data transfer. Both the browser and server need to support CORS before you can use it. The feature is compatible with all browsers. For IE, IE 10 or later is required.

The entire CORS communication process is automatically completed by the browser without any manual intervention. For developers, CORS communication and one-origin AJAX communication work in the same way and use the same code. Once the browser identifies an AJAX request for cross-origin access, it automatically adds additional header information. In some cases, an additional request is made, without being perceived by users.

Therefore, the key to CORS communication lies in the server. As long as the server implements CORS APIs, cross-origin communication can be achieved.

CORS Use Cases

Users use CORS when using a browser because access permissions are controlled by the browser instead of the server. Therefore, when other clients are used, there is no need to worry about cross-origin issues.
CORS is mainly used to directly access COS data or upload and download data on the browser side by using AJAX, without needing to go through the user's application server. For websites that use both COS and AJAX, it is recommended that CORS be used to achieve direct communication with COS.

COS Support for CORS

COS allows you to configure CORS rules to allow or deny cross-origin requests as required. The CORS rules are configured for buckets.
Whether a CORS request is allowed is independent of COS identity authentication. In other words, CORS rules of COS are only used to determine whether to add CORS-related headers. The browser determines whether to block a request. All object-related APIs and multipart-related APIs of COS support CORS-related authentication.

> Note:
>
> When two webpages that come from www.a.com and www.b.com , respectively and run in the same browser request the same cross-origin resource at the same time and the request from www.a.com arrives at the server first, the server returns the resource to the user of www.a.com with the Access-Control-Allow-Origin header. If www.b.com sends a request later, the browser returns the cached response of the last request to the user. As the header content does not match the CORS-required content, the www.b.com request fails.

CORS Configuration Example

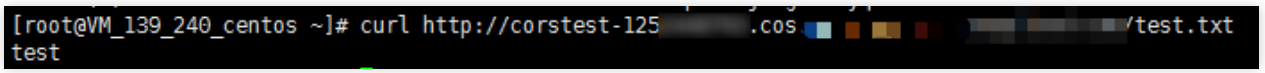
The following example shows how to configure CORS to obtain data from COS by using AJAX. The permission to the bucket used in the example is set to Public. To access a bucket with the private permission, add the signature in the request and retain the other configurations unchanged.
The bucket used in the following example is named corstest and the permission to the bucket is public read and private write.

Preparations

1. Check whether file access is normal.
   Upload a text document named test.txt to corstest. The test.txt access address is http://corstest-125xxxxxxx.cos.city.yfm4.fsphere.cn/test.txt .
   Use curl to directly access this text document. Replace the following URL with your file address:

   ```
   curl http://corstest-125xxxxxxx.cos.city.yfm4.fsphere.cn/test.txt
   ```

   Return the content of the test.txt file: test indicates that the file can be accessed normally.

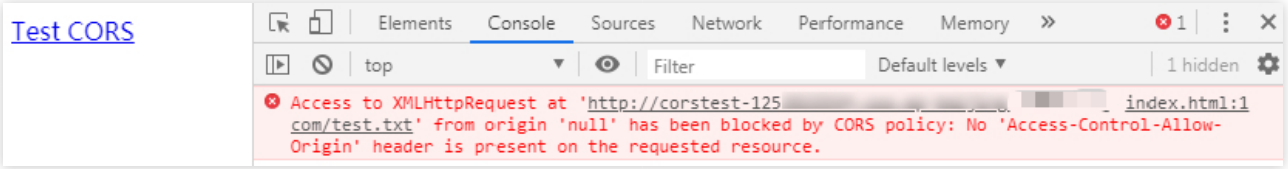


2. Access the file by using AJAX.
   The following describes how to access the test.txt file by using AJAX.

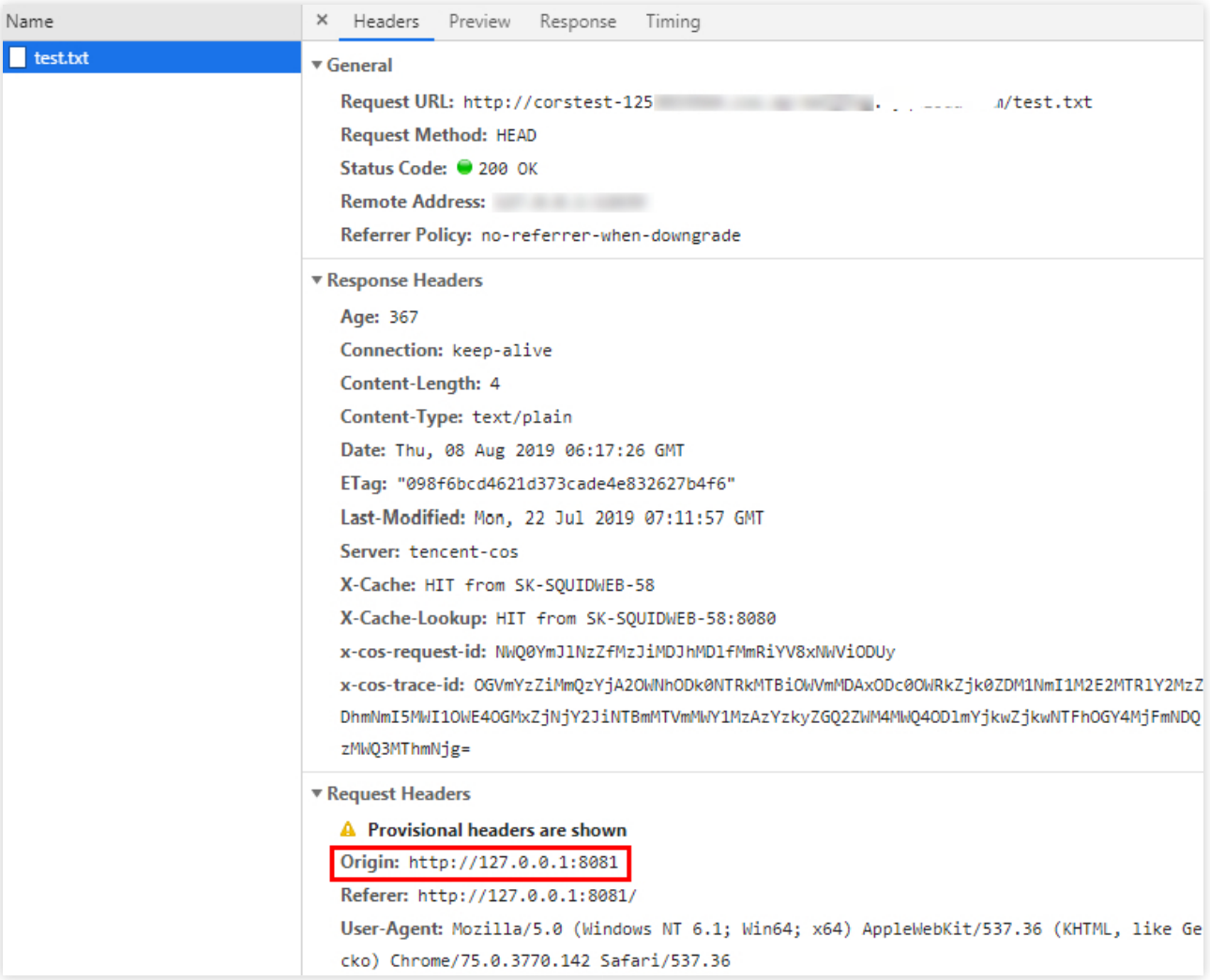
   i. Write a simple HTML file, copy the following code into it, save it as an HTML file locally, and open it with a browser. Since no custom header is set, pre-check is not required for this request.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="UTF-8">
</head>
<body>
<a href="javascript:test()">Test CORS</a>
<script>
  function test() {
    var url = 'http://corstest-125xxxxxxx.cos.city.yfm4.fsphere.cn/test.txt';
    var xhr = new XMLHttpRequest();
    xhr.open('HEAD', url);
    xhr.onload = function () {
      var headers = xhr.getAllResponseHeaders().replace(/\r\n/g, '\n');
      alert('request success, CORS allow.\n' +
        'url: ' + url + '\n' +
        'status: ' + xhr.status + '\n' +
        'headers:\n' + headers);
    };
    xhr.onerror = function () {
      alert('request error, maybe CORS error.');
    };
    xhr.send();
  }
</script>
</body>
</html>
```

   ii. Open the HTML file in the browser and click Test CORS to send a request. "No permission to access because the Access-Control-Allow-Origin header was not found." is displayed. This is because CORS is not configured for the server.



   iii. Access failed. Enter the Header page to check the cause. The browser sends a request with an Origin, indicating a cross-origin request.

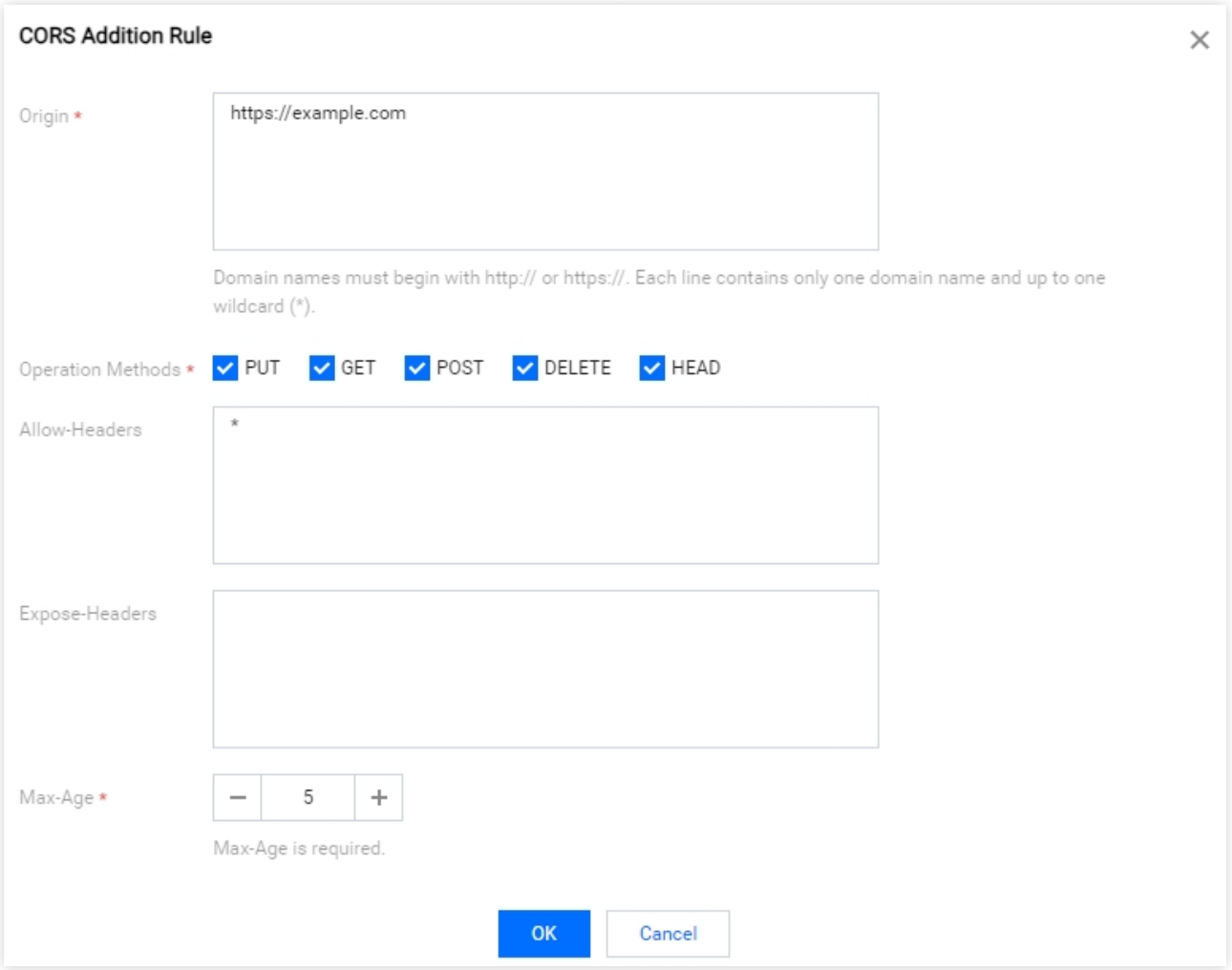


> Note:
>
> Because the webpage is set up on the server with the address http://127.0.0.1:8081 , and the Origin is http://127.0.0.1:8081 .

Configuring CORS

After the cause of the access failure is determined, you can solve the problem by configuring CORS for the bucket. In this example, CORS is configured in the COS console, which is recommended if your CORS configuration is simple.

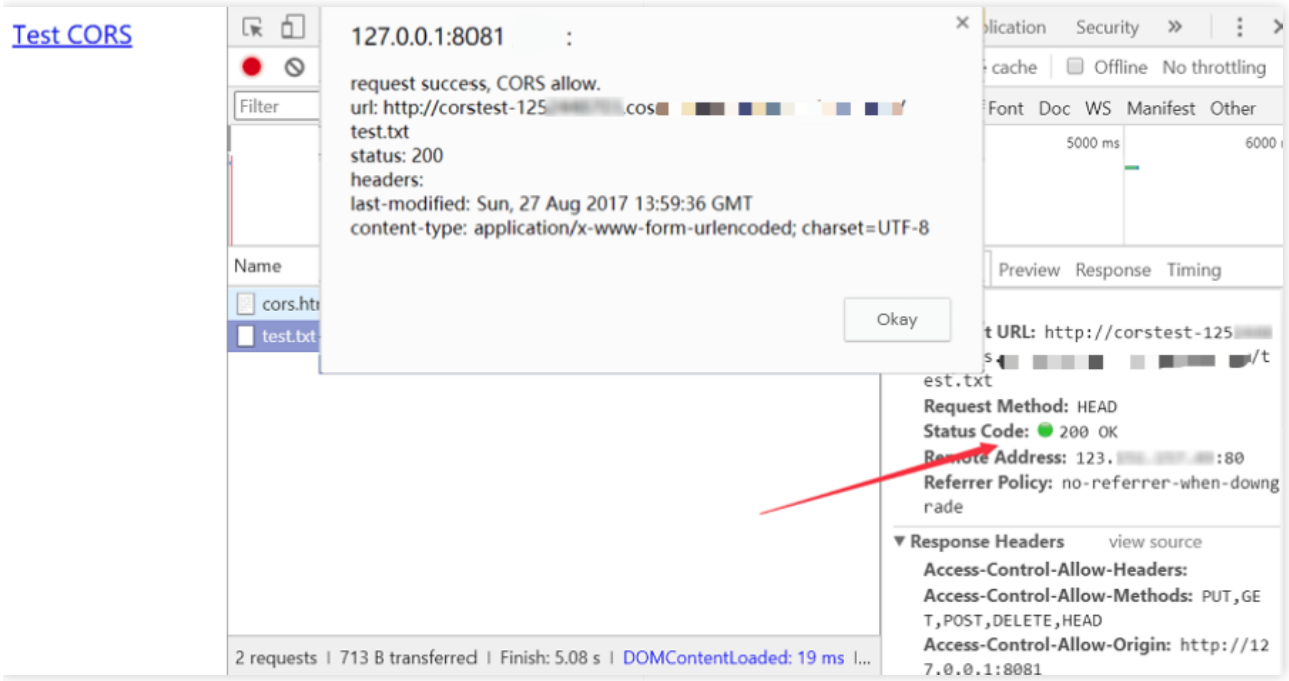
1. Log in to the COS console and click Bucket List to enter the related bucket.
2. In the left navigation tree, choose Security Management > CORS Settings to enter the cross-origin access CORS settings page.
3. Click Add Rule to add the first rule with the following least restricted configuration:



> Note:
>
> The CORS configuration consists of multiple rules, which are matched from top to bottom. The first matched rule prevails.

Verification Result

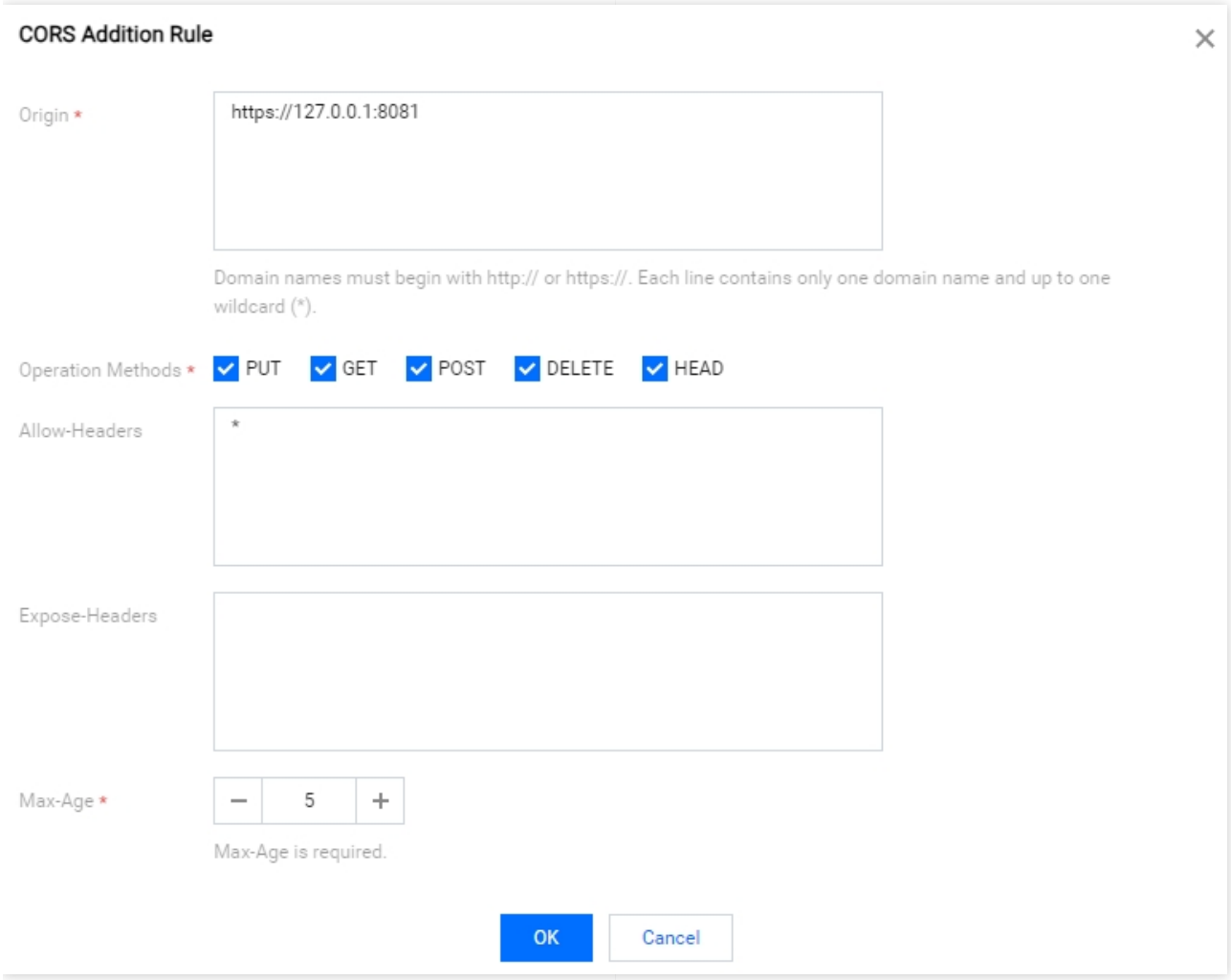
After the configuration is completed, try to access the test.txt file again. The following figure shows the result, indicating that the access is normal.



Troubleshooting and Suggestions

To avoid problems related to cross-origin access, you can set the least restricted CORS configuration as described above to allow all cross-origin requests. If an error still occurs under this configuration, the root cause may lie in other factors rather than CORS.

In addition to the least restricted configuration, you can configure a more refined control mechanism. For example, in this example, you can use the following minimum configuration to ensure a successful match:



Therefore, for most scenarios, it is recommended that you use the minimum configuration based on your usage scenario to ensure security.

CORS Configuration Items

CORS configuration items are as follows:

Origin

The sources from which cross-origin requests are allowed.

- More than one header can be specified, with one header per line.
- You can set a * , indicating that all domains are allowed, which is not recommended.
- A single specific domain name is supported, such as http://www.abc.com .
- Second-level wildcard domain names, for example, http://*.abc.com , are supported. Note that each line can contain only one asterisk * .
- Be careful not to omit the protocol name (http or https). If the port is not the default port 80, you also need to add a port.

Methods

Enumeration of one or more methods is allowed for a cross-origin request.
For example: GET, PUT, POST, DELETE, HEAD.

Allow-Headers

Header of allowed cross-origin requests.

- More than one header can be specified, with one header per line.
- Header is easily missed. Unless otherwise specified, it is recommended that it be set to *, indicating that all requests are allowed.
- There is no case sensitive.
- Each Header specified in Access-Control-Request-Headers must correspond to a value in Allowed-Header.

Expose-Headers

A list of headers exposed to the browser. That is, the response headers that you access from the application (for example, the XMLHttpRequest object of JavaScript).

- The configuration should be determined based on the requirements of the application. ETag is recommended by default.
- A wildcard is not allowed. The headers are case-insensitive, with one header per line.

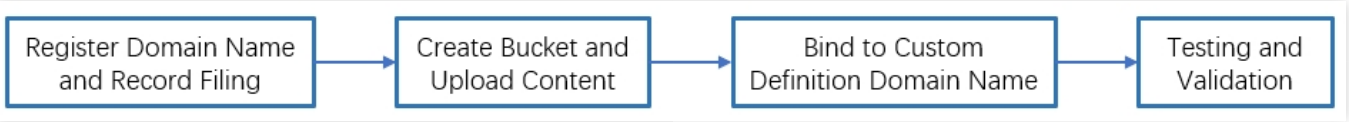
Max-Age

The time (in seconds) that the browser can cache the results of a prefetch request (OPTIONS request) for specific resources. In general cases, you can set it to a larger value, for example, 60 seconds.
This configuration item is optional.

Hosting a Static Website

Overview

This section describes how to host an existing or new static website on COS. Visitors can access the managed static website by using the static website domain name provided by COS or a bound custom domain name (such as www.example.com). The following figure shows the detailed steps for hosting a static website on COS.



> Note:
>
> After enabling the static website configuration, you need to use the static website domain name to access the COS origin server for it to take effect. If you use the COS default domain name to access, there is no static website effect.

Preparation

The following are related services that may be used in the practice process:

- Domains (optional): Before hosting a static website, you need to have a domain, such as www.example.com . If you are using a static website domain provided by COS, you do not need to use this service.
- COS: A COS bucket is created to store the uploaded web page content.
- CDN: The CDN and DNS resolution services are used to bind the domain name to the website content, accelerate static website access, reduce the access delay, and improve the browsing experience.
- DNSPod: DNS resolution is used to ensure that a custom domain name is used to access the static website.

> Note:
>
> All steps in this guide use www.example.com as an example domain. In actual operation, replace this domain with your domain.

Step 1: Domain Registration and Filing (Optional)

Domain registration is the basis for establishing any service on the Internet. After registering a domain, you need to file it before the website can be accessed normally. Proceed according to your specific situation:

- If the domain has been registered and filed, you can skip this step and proceed to Step 2.

- If the domain name is registered but not filed, proceed with Domain Name Filing.
- If the domain is not registered, first Register Domain Name and proceed with Domain Name Filing.

> Note:
>
> If you are using a static website domain provided by COS, you can skip this step.

step 2: create a bucket and upload content">Step 2: Create a Bucket and Upload Content

After completing domain registration and filing, you need to perform the following tasks in the COS console to create and configure website content:

- Creating a bucket
- Configure the bucket and upload content

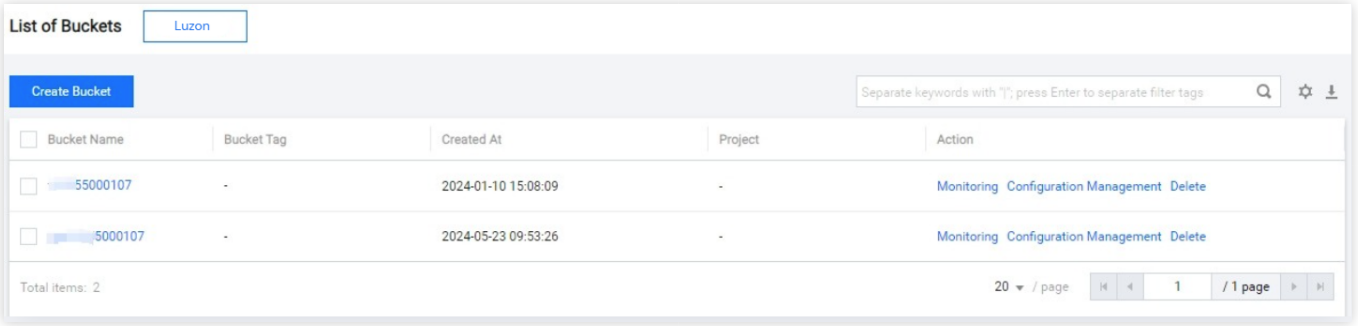
1. creating a bucket">1. Creating a Bucket

Use your account to log in to the COS console and create a corresponding bucket for your website. Buckets are used to store data. You can store website content in a bucket.

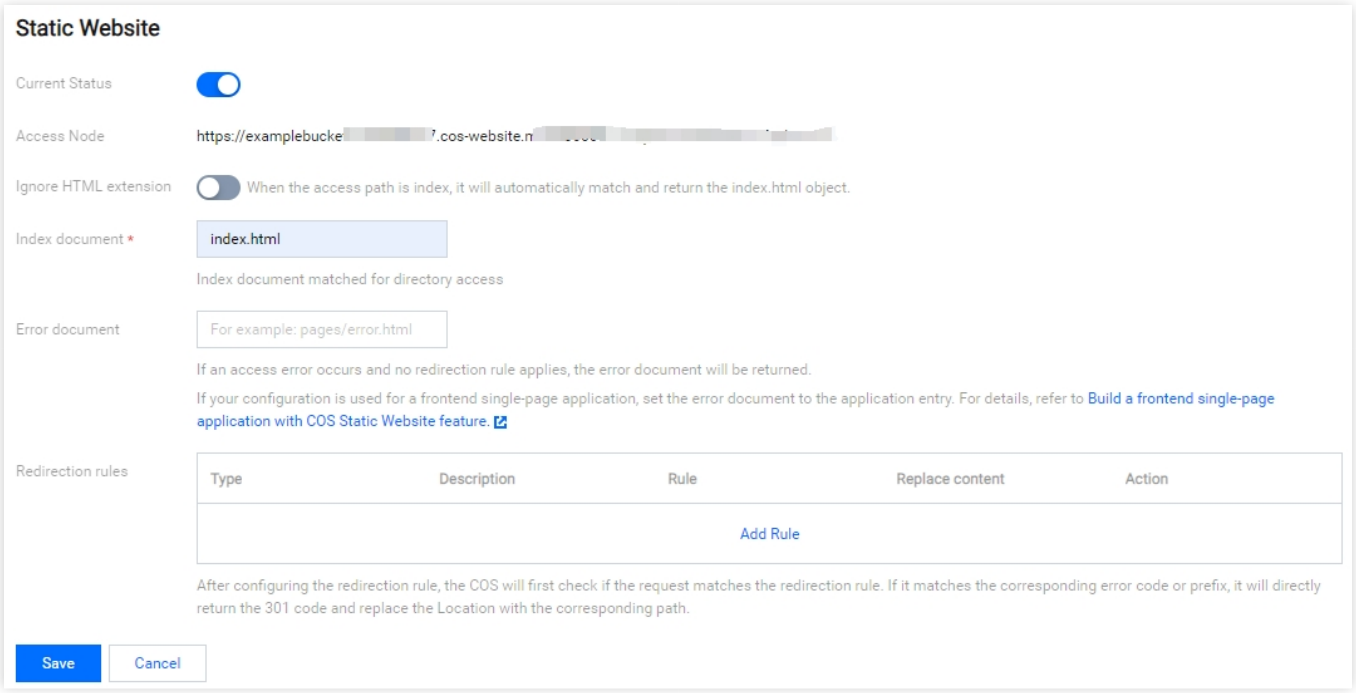
If you use COS for the first time, you can directly create a bucket by clicking Create Bucket on the overview page or Bucket List in the left navigation tree in the console. For detailed steps, see Create Bucket.

2. configuring the bucket and uploading content">2. Configuring the Bucket and Uploading Content

1. Enable the Static Website setting for the bucket by:
   Log in to the COS console. In the left-side menu bar, click Bucket List, find the created bucket, and click Configuration Management on the right.



   In the left-side menu bar, choose Basic Configuration > Static Website, click Modify, set the status to enabled, and set the index document to index.html. Leave the rest unconfigured, and click Save.



2. Upload your website content to the created bucket. For detailed steps, see Upload Object.
   The content stored in the bucket can be text files, photos, videos, and anything you want to host. If you have not built a website yet, you only need to create one file according to this practice.
   For example, you can use the following HTML to create a file and upload it to your bucket. In a later step, you will provide this file as the index documentation for your site.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello COS!</title>
    <meta charset="utf-8">
  </head>
  <body>
    <p>Welcome to use the static website feature of COS.</p>
    <p>This is the homepage!</p>
  </body>
</html>
```

> Note:
>
> After the static website feature is enabled, when a user accesses any first-level directory without a file pointer, COS will prioritize matching index.html in the corresponding bucket directory by default, followed by index.htm. If there is no such file, a 404 error is returned.

Data Security
Anti-Hotlinking Practice

Feature Introduction

COS allows you to configure hotlink protection. You can set a blocklist and allowlist through hotlink protection in the console to ensure security.

How Does Hotlink Protection Work

Hotlink protection works by checking the Referer field in the request header:

- The Referer field is a part of the header. When a browser sends a request to a web server, the request usually carries the Referer field to tell the server from which page the request comes. In this way, the server can prohibit or allow websites from some sources to access resources.
- If you open the file link https://examplebucket-1250000000.cos.city.yfm4.fsphere.cn/1.jpg directly in a browser, the Referer field is not carried in the request header.
  For example, in the following figure, 1.jpg is embedded in http://127.0.0.1/test/test.html , and the Referer field pointing to the access source is carried for access to https://127.0.0.1/test/test.html .

Hotlink Protection Case Analysis

User A uploads the image resource 1.jpg to COS, and the accessible link to the image is https://examplebucket-1250000000.cos.city.yfm4.fsphere.cn/1.jpg .

- User A embeds the image in his/her webpage https://example.com/index.html , and the image is accessible.
- User B sees the image on user A's webpage and decides to embed it in his/her webpage https://b.com/test/test.html , and the image is properly displayed on user B's webpage.
  In the above case, user A's image resource 1.jpg was hotlinked by user B. Resources in COS are being used by user B's webpage without being known by user A, and user A suffers from losses caused by extra traffic fees.

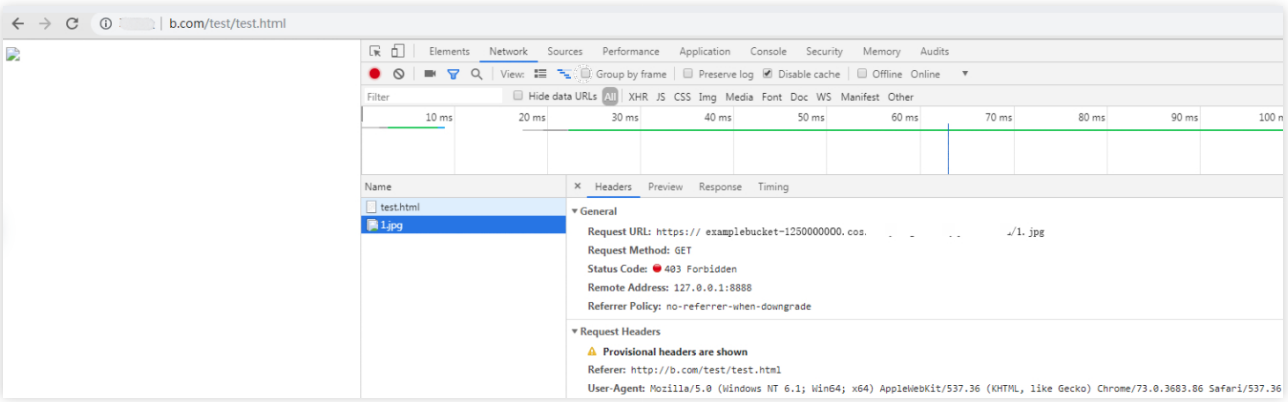
Solution

Based on the above hotlink protection case analysis, user A can prevent user B from hotlinking his/her images by setting hotlink protection in the following ways:

1. Set a hotlink protection rule for bucket examplebucket-1250000000. There are two methods for preventing user B from hotlinking:
   - Method 1: Configure a blocklist, set the domain name to *.b.com , and save the configuration for it to take effect.
   - Method 2: Configure an allowlist, set the domain name to *.example.com , and save the configuration for it

to take effect.

2. After hotlink protection is enabled:
   - The image is displayed properly when http://www.example.com/index.html is accessed.
   - The image cannot be displayed when https://b.com/test/test.html is accessed, as shown below.



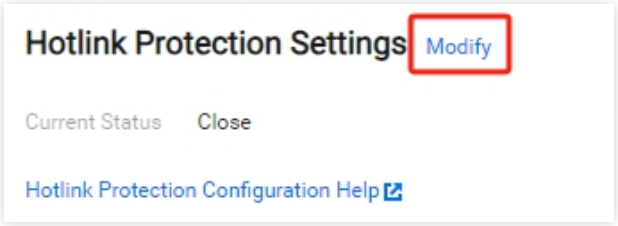
Directions

1. Log in to the COS console and select Bucket List from the left sidebar to enter the bucket list page.

2. Select the bucket for which hotlink protection is configured and enter the file list page of the bucket.

3. In the left navigation tree, choose Security Management > Hotlink Protection Settings to enter the hotlink protection settings page.

4. Click Modify to enter the editable status.



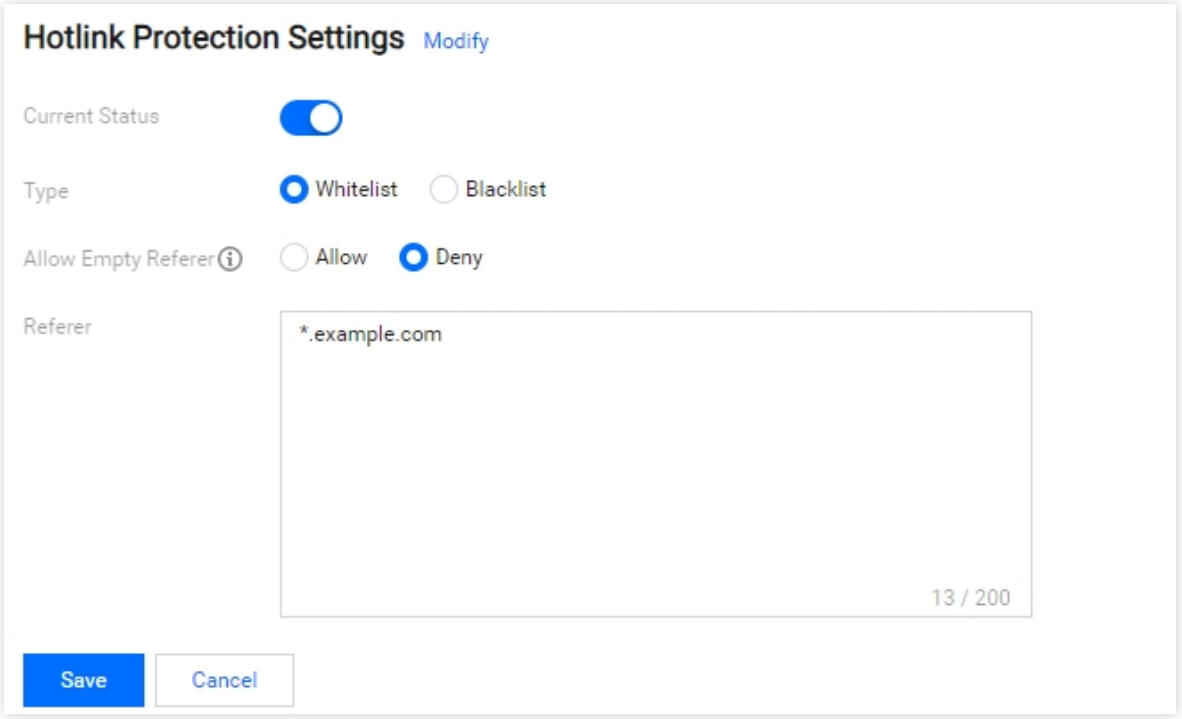
5. Enable hotlink protection and configure the list type and domain name. Method 2 is selected here. The detailed description is as follows:
   - Type: There are two types of blocklist and allowlist:
     - Blocklist: Prohibits domain names in the list to access the default access address of the bucket. If a domain name in the list accesses the default access address of the bucket, a 403 error will be returned.
     - Allowlist: Prohibits domain names not in the list to access the default access address of the bucket. If

a domain name not in the list accesses the default access address of the bucket, a 403 error will be returned.

- Referer: Setting the domain supports up to ten domain names and prefix matching, and addresses in the form of domain names, IP addresses, and wildcard characters * are supported. One address occupies one line. If there are multiple addresses, change the line. Configuration rule descriptions and examples are as follows:
  - Domain names and IPs with ports are supported, such as www.example.com:8080 and 10.10.10.10:8080 .
  - When configure *.example.com , you can hit address such as a.example.com/123 and other address prefixed with example.com .
  - When configure example.com , you can hit address prefixed with https://example.com and http://example.com .
  - When configure example.com , you can hit its port domain example.com:8080 .
  - When configure example.com:8080 , the domain example.com will not be hit.
  - When configure *.example.com , you can limit its second-level and third-level domain example.com , b.example.com , and abexample.com .

> Note
>
> After the status of hotlink protection is set to enabled, enter the corresponding domain name.

6. After the configuration is completed, click Save.



Data Verification
MD5 Verification

Introduction

Errors may occur during data transmission between the client and the server. Cloud Object Storage (COS) can ensure the integrity of uploaded data through MD5 verification. Only when the MD5 checksum of the data received by the COS server matches that set by the user can the data be uploaded successfully.
Each object in COS corresponds to an ETag, which serves as an information identifier for the object's content at the time of creation. However, the ETag may not necessarily be equivalent to the MD5 checksum of the object's content. Therefore, it cannot be used to verify whether a downloaded object is consistent with the original one. Instead, users can utilize custom object metadata (x-cos-meta-*) to perform consistency checks between the downloaded object and the original.

Data Verification Method

- Verifying the uploaded object
- If users need to verify whether an object uploaded to COS matches the local object, they can set the Content-MD5 field in the HTTP request to the Base64-encoded MD5 checksum of the object's content during upload. In this case, the COS server will verify the uploaded object. The object will only be uploaded successfully if the MD5 checksum received by the COS server matches the user-set Content-MD5 value.
- Verifying the downloaded object
  If users need to verify whether the downloaded object matches the original object, they can calculate the object checksum using a verification algorithm during upload, and set the checksum via custom metadata. After downloading the object, users can recalculate the checksum and compare it with the custom metadata for verification. In this approach, users can freely choose the verification algorithm, but for the same object, the algorithm used during upload and download should remain consistent.

API Example

Simple Upload Request

Below is an example request for a user to upload an object. When uploading an object, the user sets the Content-MD5 to the Base64-encoded MD5 checksum of the object content. In this case, the object can be uploaded successfully only when the MD5 checksum of the object received by the COS server matches the

Content-MD5 set by the user. Additionally, the custom metadata x-cos-meta-md5 should be set to the checksum value of the object.

> Note:
>
> The example demonstrates obtaining the checksum of the object using the MD5 verification algorithm. Users may choose alternative verification algorithms according to their needs.

```
PUT /exampleobject HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Fri, 21 Jun 2019 09:24:28 GMT
Content-Type: image/jpeg
Content-Length: 13
Content-MD5: ti4QvKtVqIJAvZxDbP/c+Q==
Authorization: q-sign-algorithm=sha1&q-ak=**********************************&q-sign-time=156110
9068;1561116268&q-key-time=1561109068;1561116268&q-header-list=content-length;content-md
5;content-type;date;host&q-url-param-list=&q-signature=****************************************
x-cos-meta-md5: b62e10bcab55a88240bd9c436cffdcf9
Connection: close

[Object Content]
```

Multipart Upload Request

Below is an example request for initializing a multipart upload. When uploading object parts, users can set custom metadata for the object through multipart upload initialization, for example, setting the custom metadata x-cos-meta-md5 as the object's checksum.

```
POST /exampleobject?uploads HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Fri, 21 Jun 2019 09:45:12 GMT
Authorization: q-sign-algorithm=sha1&q-ak=**********************************&q-sign-time=156110
9068;1561116268&q-key-time=1561109068;1561116268&q-header-list=content-length;content-md
5;content-type;date;host&q-url-param-list=&q-signature=****************************************
x-cos-meta-md5: b62e10bcab55a88240bd9c436cffdcf9
```

> Note:
>
> For files uploaded via multipart upload, COS only verifies the MD5 value of each part and does not calculate the MD5 value of the complete file composed of merged parts.

Object Download Response

Below is an example response received after a user sends a request to download an object. From the response, users can obtain the custom metadata x-cos-meta-md5 of the object. By recalculating the

checksum of the downloaded object and comparing it with this custom metadata, users can verify whether the downloaded object is consistent with the original object.

```
HTTP/1.1 200 OK
Content-Type: application/octet-stream
Content-Length: 13
Connection: close
Accept-Ranges: bytes
Cache-Control: max-age=86400
Content-Disposition: attachment; filename=example.jpg
Date: Thu, 04 Jul 2019 11:33:00 GMT
ETag: "b62e10bcab55a88240bd9c436cffdcf9"
Last-Modified: Thu, 04 Jul 2019 11:32:55 GMT

x-cos-request-id: NWQxZGUzZWNfNjI4NWQ2NF9lMWYyXzk1NjFj****
x-cos-meta-md5: b62e10bcab55a88240bd9c436cffdcf9

[Object Content]
```

Accessing COS with the AWS S3 SDK

Overview

COS provides APIs compatible with AWS S3. After your data is migrated from AWS S3 to COS, your client can easily be compatible with COS services through simple configuration modification. This section describes how to adapt to S3 SDKs of different development platforms. After adaptation, you can use S3 SDK APIs to access files in COS.

Preparations

1. You have registered an account and obtained the SecretID and SecretKey from Cloud Access Management > Cloud API Key.
2. You have a client that has been integrated with the S3 SDK and runs properly.

Java

The following example describes how to adapt to AWS Java SDK 2.20.0 to access COS services.

1. Modifying the AWS Configuration and Certificate Files

> Note:
>
> The following example describes how to modify AWS configuration and certificate files on Linux.

The default configuration file of the AWS SDK is typically located under the user directory. For details, see Configuration and Certificate Files.

- Add the following configuration information to the configuration file (located in ~/.aws/config):

```
[default]
s3 =
addressing_style = virtual
```

- Configure the key in the certificate file (located in ~/.aws/credentials).

```
[default]
aws_access_key_id = [COS_SECRETID]
aws_secret_access_key = [COS_SECRETKEY]
```

2. Setting the Endpoint in the Code

The bucket region ap-region-jcctest-ops is used as an example. The example code is as follows:

```
S3Client s3 = S3Client.builder()
        .endpointOverride(URI.create("http://cos.ap-region-jcctest-ops.cospub.yfm13.fsphere.cn"))
        .region(Region.of("ap-region-jcctest-ops"))
        .build();
```

Python

The following example describes how to adapt to AWS Python SDK 1.9.205 to access COS services.

1. Modifying the AWS Configuration and Certificate Files

> Note:
>
> The following example describes how to modify AWS configuration and certificate files on Linux.

The default configuration file of the AWS SDK is typically located under the user directory. For details, see Configuration and Certificate Files.

- Configure the key in the certificate file (located in ~/.aws/credentials).

```
[default]
aws_access_key_id = [COS_SECRETID]
aws_secret_access_key = [COS_SECRETKEY]
```

2. Setting the Endpoint in the Code

The bucket region ap-region-jcctest-ops is used as an example.

```
s3 = boto3.client(
    's3',
    region_name='ap-region-jcctest-ops',
    endpoint_url='http://cos.ap-region-jcctest-ops.cospub.yfm13.fsphere.cn',
    verify=True,    # Whether to verify the certificate. True/False/'/path/to/ca-bundle.crt'
    use_ssl=False   # Whether to use SSL/TLS.
)
```

Go

The following example describes how to adapt to AWS Go SDK v1.55.6 to access COS services.

1. Modifying the AWS Certificate File

- Configure the key in the certificate file (located in ~/.aws/credentials).

```
[default]
aws_access_key_id = [COS_SECRETID]
aws_secret_access_key = [COS_SECRETKEY]
```

2. Simple Examples

```
package main

import (
    "fmt"
    "log"
    "os"

    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/s3"
)

func main() {
    // 1. Create AWS session
    // Credentials are loaded from environment variables or ~/.aws/credentials
    region := "ap-region-jcctest-ops"
    endpoint := "http://cos.ap-region-jcctest-ops.cospub.yfm13.fsphere.cn"

    sess, err := session.NewSession(&aws.Config{
        Region:   aws.String(region),
        Endpoint: &endpoint,
    })
    if err != nil {
        log.Fatal("Failed to create session:", err)
    }

    // 2. Create S3 service client
```

```
svc := s3.New(sess)

// 3. List all buckets
fmt.Println("Listing buckets:")
result, err := svc.ListBuckets(nil)
if err != nil {
    log.Fatal("Failed to list buckets:", err)
}

for _, b := range result.Buckets {
    fmt.Printf("* %s created at %s\n",
        aws.StringValue(b.Name), aws.TimeValue(b.CreationDate))
}

// 4. Upload file to S3
bucketName := "test-1255000327"            // Replace with your bucket name
fileName := "/tmp/test.txt"                // Local file name
objectKey := "uploaded_test_by_aws_go_sdk.txt" // Object key in S3

file, err := os.Open(fileName)
if err != nil {
    log.Fatal("Failed to open file:", err)
}
defer file.Close()

_, err = svc.PutObject(&s3.PutObjectInput{
    Bucket: aws.String(bucketName),
    Key:    aws.String(objectKey),
    Body:   file,
})
if err != nil {
    log.Fatal("Failed to upload file:", err)
}
fmt.Printf("Successfully uploaded file %s to %s/%s\n", fileName, bucketName, objectKey)

// 5. Download file from S3
downloadedFile := "downloaded_test.txt"
out, err := os.Create(downloadedFile)
if err != nil {
    log.Fatal("Failed to create download file:", err)
}
defer out.Close()

result2, err := svc.GetObject(&s3.GetObjectInput{
    Bucket: aws.String(bucketName),
    Key:    aws.String(objectKey),
})
```

```
    if err != nil {
        log.Fatal("Failed to download file:", err)
    }
    defer result2.Body.Close()

    _, err = out.ReadFrom(result2.Body)
    if err != nil {
        log.Fatal("Failed to save downloaded file:", err)
    }
    fmt.Printf("Successfully downloaded file %s/%s to %s\n", bucketName, objectKey, downloadedFile)
}
```

C++

The following example describes how to adapt to AWS C++ SDK 1.7.68 to access COS services.

1. Modifying the AWS Configuration and Certificate Files

> Note:
>
> The following example describes how to modify AWS configuration and certificate files on Linux.

The default configuration file of the AWS SDK is typically located under the user directory. For details, see Configuration and Certificate Files.

- Add the following configuration information to the configuration file (located in ~/.aws/config):

```
[default]
s3 =
addressing_style = virtual
```

- Configure the key in the certificate file (located in ~/.aws/credentials).

```
[default]
aws_access_key_id = [COS_SECRETID]
aws_secret_access_key = [COS_SECRETKEY]
```

2. Setting the Endpoint in the Code

The bucket region ap-region-jcctest-ops is used as an example. The example code is as follows:

```
Aws::Client::ClientConfiguration awsCC;
awsCC.scheme = Aws::Http::Scheme::HTTP;
awsCC.region = "ap-region-jcctest-ops";
awsCC.endpointOverride = "cos.ap-region-jcctest-ops.cospub.yfm13.fsphere.cn";
Aws::S3::S3Client s3_client(awsCC);
```

API References
Overview

Description

COS uses XML APIs, which are lightweight, connectionless, and stateless. By calling XML APIs, you can send requests and receive responses directly over HTTP/HTTPS to interact with the COS backend.

Before consulting other API documents, please read the signature authentication in detail first.

Basic Info

This section introduces the main concepts and terms that are essential for using COS more effectively.

Name	Description
AppID	User-dimensional unique resource identifier of a developer when accessing COS services, which is used to identify resources.
SecretID	SecretID is the project identity identification ID owned by the developer, used for identity authentication.
SecretKey	SecretKey is the project identity key owned by the developer.
Bucket	A bucket is a container used to store data in COS, which is the first-level directory stored under Appid by the user, and each object is stored in a bucket.
Object	An object is a specific file stored in COS and is the basic entity of storage.
Region	Region information in the domain name, enumerated values: reg1 (Region1)

Getting Started

To use COS APIs, perform the following steps:

1. Obtain the AppID, SecretID, and SecretKey.
2. Write a signature authentication algorithm program (or use any server-side SDK).
3. Calculate the signature and call the APIs to perform operations.

Common Request Headers

Description

This document will introduce you to the common request headers that will be used when using the API. The headers mentioned below will not be reiterated in the specific API documents later.

Request Header List

Header Name	Description	Type	Required
Authorization	Signature information that carries authentication information and is used to verify the validity of a request. For publicly readable files, this header is not needed.	String	No
Content-Length	The HTTP request content length (in bytes) defined in RFC 2616, commonly used for operations of the PUT type API.	String	No
Content-Type	The HTTP request content type (MIME) defined in RFC 2616, such as text/plain.	String	No
Content-MD5	Base64-encoded 128-bit content MD5 checksum as defined in RFC 1864. This header is used to verify whether the file content has changed.	String	No
Date	GMT time defined in RFC 1123, for example, Wed, 30 Mar. 2016 23:00:00 GMT.	String	No
Expect	When using Expect: 100-continue, the request content is sent only after receiving confirmation from the server. This option can be used to validate headers without sending data content. Valid value: 100-continue.	String	No
Host	The host of the request, in the form of ..	String	Yes

Server-Side Encryption Dedicated Header

For APIs supporting server-side encryption (SSE), the following request headers are applicable based on different encryption methods. Please refer to specific API documents to determine if SSE is applicable. The whether these headers are required only applies to scenarios using SSE. If the request does not support SSE APIs or does not use SSE, these headers do not need to be included. For details, please refer to Server-side

Encryption Overview.

SSE-COS

Header Name	Description	Type	Required
x-cos-server-side-encryption	SSE algorithm, which is set to AES256 when SSE-COS is used.	string	If this header is carried during object upload or copying (including simple upload/copy and multipart upload/copy), SSE-COS is used. If this header is not carried, encryption is performed based on the bucket encryption configuration. This header should not be specified when objects are downloaded.

SSE-KMS

Header Name	Description	Type	Required
x-cos-server-side-encryption	SSE algorithm, which is set to cos/kms when SSE-KMS is used.	string	If this header is carried during object upload or copying (including simple upload/copy and multipart upload/copy), SSE-KMS is used. If this header is not carried, encryption is performed based on the bucket encryption configuration. This header should not be specified when objects are downloaded.
x-cos-server-side-encryption-cos-kms-key-id	CMK of the KMS when x-cos-server-side-encryption is set to cos/kms. If it is not specified, the CMK created by COS by default is used. For details, see SSE-KMS Encryption.	string	No
x-cos-server-side-encryption-context	Encryption context when x-cos-server-side-encryption is set to cos/kms. The value is the Base64-encoded key-value pair of the encryption context in JSON format, for example, eyJhIjoiYXNkZmEiLCJiIjoiMTIzMzIxIn0= .	string	No

SSE-C

Header Name	Description	Type	Required
x-cos-server-side-encryption-customer-algorithm	Server-side encryption algorithm, currently only supports AES256.	string	Yes
x-cos-server-side-encryption-customer-key	Base64-encoded SSE key, for example, MDEyMzQ1Njc4OUFCQ0RFRjAxMjM0NTY3ODlBQkNERUY= .	string	Yes
x-cos-server-side-encryption-customer-key-MD5	Base64-encoded MD5 hash value of the SSE key, for example, U5L61r7jcwdNvT7frmUG8g== .	string	Yes

Public Response Headers

Description

This document describes common response headers that may be used in API requests. The headers mentioned below will not be addressed again in related API documents.

Response Headers

Header Name	Description	Type
Content-Length	HTTP response length in bytes as defined in RFC 2616.	String
Content-Type	HTTP response type (MIME) as defined in RFC 2616.	String
Connection	Network connection status between the client and the server. Enumerated values: keep-alive and close.	Enum
Date	Server response time in GMT format as defined in RFC 1123.	String
Etag	An entity tag (ETag) identifies the object content upon creation. The value of ETag is not necessarily the MD5 checksum of the object according to different request situations. The value of ETag can be used to check whether the object content is changed.	String
x-cos-request-id	An ID automatically generated by the server for each request when it is sent.	String
x-cos-trace-id	An ID automatically generated by the server for each error when a request fails.	String

Server-Side Encryption Dedicated Header

For APIs that support Server Side Encryption (SSE) and use SSE in requests, response headers described in the following are returned based on the specific encryption type. For details, see Server-side Encryption Overview.

SSE-COS

Header Name	Description	Type
x-cos-server-side-encryption	When SSE-COS is used during object upload or objects encrypted with SSE-COS are downloaded, responses to the request carry this header, indicating that the SSE algorithm is used during object upload.	string

SSE-KMS

Header Name	Description	Type
x-cos-server-side-encryption	When SSE-KMS is used during object upload or objects encrypted with SSE-KMS are downloaded, responses to the request carry this header, indicating that the SSE algorithm is used during object upload.	string
x-cos-server-side-encryption-cos-kms-key-id	The customer master key (CMK) of the Key Management Service (KMS) is returned. If it is not specified, the CMK created by COS by default is returned.	string

SSE-C

Header Name	Description	Type
x-cos-server-side-encryption-customer-algorithm	When SSE-C is used during object upload or objects encrypted with SSE-C are downloaded, responses to the request carry this header, indicating that the SSE algorithm is used during object upload.	string
x-cos-server-side-encryption-customer-key-MD5	Base64-encoded MD5 hash value of the SSE key used during object upload. For example, U5L61r7jcwdNvT7frmUG8g== .	string

Error Code

Overview

This section describes the error codes and error information returned when requests fail.

Error Response

Content-Type: application/xml

HTTP status codes are 3XX, 4XX, and 5XX. Specifically, for the PUT Object - Copy API, the response body may contain errors even if the HTTP status code is 200.

Response Body

```
<?xml version='1.0' encoding='utf-8' ?>
<Error>
   <Code>string</Code>
   <Message>string</Message>
   <Resource>string</Resource>
   <RequestId>string</RequestId>
   <TraceId>string</TraceId>
</Error>
```

Specific node descriptions are as follows:

Node Name (Keyword)	Parent Node	Description	Type
Error	No	Contains all error information.	Container

Content of Error in the Container Node:

Node Name (Keyword)	Parent Node	Description	Type
Code	Error	Error code, which is used to locate the unique error condition and determine the error scenario. For specific error codes, see the following section.	string
Message	Error	Detailed error information.	string
Resource	Error	Requested resource, which can be a bucket or object URL.	string

Node Name (Keyword)	Parent Node	Description	Type
RequestId	Error	ID automatically generated by the server for each request. The ID can assist COS in locating encountered problems quickly.	string
TraceId	Error	ID automatically generated by the server for each request error. The ID can assist COS in locating encountered problems quickly.	string

Error Code List

3XX Errors

Error Code	Description	HTTP Status Code
PermanentRedirect	The resource location is changed permanently. Use the HTTP Location response header to redirect to the correct new location.	301 Moved Permanently
TemporaryRedirect	The resource location is changed temporarily. Use the HTTP Location response header to redirect to the correct new location.	302 Moved Temporarily
Redirect	Temporary redirection is performed.	307 Moved Temporarily
TemporaryRedirect	During the DNS update period, you are temporarily redirected.	307 Moved Temporarily

4XX Errors

Error Code	Description	HTTP Status Code
AppendPositionErr	The object length does not match position during the append operation.	400 Bad Request
AttachmentFull	The number of ACLs and policies reached the limit.	400 Bad Request

Error Code	Description	HTTP Status Code
BadDigest	The provided Content-MD5 value is inconsistent with the MD5 hash value in the request body received by the server.	400 Bad Request
EntityTooLarge	The size of the uploaded object exceeds the specified maximum value.	400 Bad Request
EntityTooSmall	The size of the uploaded object is less than the specified minimum value, commonly seen in multipart uploads.	400 Bad Request
IncorrectNumberOfFilesInPostRequest	Only one object can be uploaded at a time in a POST request.	400 Bad Request
InvalidArgument	The request parameter is invalid.	400 Bad Request
InvalidBucketName	The bucket name is invalid.	400 Bad Request
InvalidCopySource	Object source copying is invalid.	400 Bad Request
InvalidDigest	The given Content-MD5 value is invalid.	400 Bad Request
InvalidPart	Parts are missing.	400 Bad Request
InvalidPartOrder	The numbers of parts for the multipart upload are not sequential.	400 Bad Request
InvalidRegionName	The region name is invalid.	400 Bad Request
InvalidRequest	The request is invalid.	400 Bad Request
InvalidSHA1Digest	The request fails the SHA-1 verification.	400 Bad Request
InvalidURI	The URI is invalid.	400 Bad Request
KeyTooLong	The object key is too long.	400 Bad Request
LifeCycleIdNotUnique	The lifecycle ID is not unique.	400 Bad Request
LifeCycleRuleConflicted	Lifecycle setting conflicts exist.	400 Bad Request
MalformedPOSTRequest	The body content of the POST request is invalid.	400 Bad Request
MalformedXML	The XML format of the request body does not comply with the XML syntax.	400 Bad Request

Error Code	Description	HTTP Status Code
MissingAppid	APPID is missing in the request header.	400 Bad Request
MissingContentMD5	Content-MD5 is missing in the request header.	400 Bad Request
MissingHost	Host is missing in the request header.	400 Bad Request
MissingRequestBodyError	The request body is missing.	400 Bad Request
MultiBucketNotSupport	Only one destination bucket can be set for cross-region replication.	400 Bad Request
NoSuchVersion	The specified version does not exist.	400 Bad Request
NotSupportedStorageClass	The specified storage type is not supported.	400 Bad Request
ObjectNotAppendable	The specified object cannot be appended.	400 Bad Request
PolicyFull	The number of ACLs and policies reached the limit.	400 Bad Request
RequestTimeOut	Data reading times out. Check whether the network speed is slow or the upload concurrency is too large.	400 Bad Request
TooManyBuckets	The number of buckets exceeds the upper limit of 200.	400 Bad Request
UnexpectedContent	The request does not support the related content.	400 Bad Request
VerifyAlgorithmNotSupported	The verification algorithm is not supported.	400 Bad Request
WebsiteURLInvalid	The custom domain name URL is invalid.	400 Bad Request
XMLSizeLimit	The XML length exceeds the limit.	400 Bad Request
AccessDenied	The signature or permission is incorrect, and the access is denied.	403 Forbidden
ExpiredToken	The signature string has expired.	403 Forbidden
InvalidAccessKeyId	The SecretID does not exist.	403 Forbidden

Error Code	Description	HTTP Status Code
InvalidObjectState	The request content conflicts with the object attributes.	403 Forbidden
InvalidObjectStorage	The storage type is invalid.	403 Forbidden
RequestTimeTooSkewed	The difference between the local time and server time is too large, exceeding 15 minutes.	403 Forbidden
SignatureDoesNotMatch	The signature calculated by the client does not match the signature calculated by the COS server.	403 Forbidden
NoSuchBucket	The specified bucket does not exist.	404 Not Found
NoSuchCopySource	The copied object source does not exist.	404 Not Found
NoSuchCORSConfiguration	The specified CORS settings do not exist.	404 Not Found
NoSuchKey	The specified object key does not exist.	404 Not Found
NoSuchLifecycleConfiguration	The specified lifecycle settings do not exist.	404 Not Found
NoSuchTagSet	The specified tag collection does not exist.	404 Not Found
NoSuchUpload	The specified UploadId for multipart upload does not exist.	404 Not Found
NoSuchWebsiteConfiguration	The static website configuration does not exist.	404 Not Found
MethodNotAllowed	The resource does not support this HTTP method.	405 Method Not Allowed
RestoreNonArchiveObject	Non-ARCHIVED objects cannot be restored.	405 Method Not Allowed
BucketAlreadyExists	The specified bucket already exists.	409 Conflict
BucketAlreadyExistsDiffRegion	The specified bucket already exists in another region.	409 Conflict
BucketAlreadyOwnedByYou	The specified bucket already exists.	409 Conflict

Error Code	Description	HTTP Status Code
BucketNotEmpty	The bucket is not empty.	409 Conflict
InvalidBucketState	The bucket status conflicts with the operation request, such as conflicts between multi-version management and cross-region replication.	409 Conflict
PathConflict	Millisecond-level concurrency conflict occurs for objects with the same name.	409 Conflict
RestoreAlreadyInProgress	The object is being recovered.	409 Conflict
MissingContentLength	The Content-Length request header is missing.	411 Length Required
PreconditionFailed	Precondition matching failed.	412 Precondition
InvalidRange	The requested object range is invalid.	416 Requested Range Not Satisfiable
UnavailableForLegalReasons	Service is unavailable due to legal reasons.	451 Unavailable For Legal Reasons

5XX Errors

Error Code	Description	HTTP Status Code
InternalErrror	An internal server error occurs.	500 Internal Server
NotImplemented	The request is not implemented.	501 Not Implemented
ServiceUnavailable	The service is temporarily unavailable. Please try again.	503 Service Unavailable
SlowDown	The access frequency needs to be reduced.	503 Slow Down

Request Signing

> Note:
>
> 1. This document applies only to the COS XML version.
> 2. This document does not apply to HTTP requests for POST objects.

When using COS, you can initiate HTTP anonymous requests or HTTP signed requests to COS through the RESTful API. For signed requests, the COS server verifies the identity of the requester.

- Anonymous Request: An HTTP request that doesn't carry any identity information or authentication credentials, performed through RESTful API operations.
- Signed request: An HTTP request is added with a signature when it is sent. After receiving the message, the COS server verifies the identity. If verification is successful, it accepts and executes the request. Otherwise, it returns an error information and discards the request.

COS verifies the identity based on a custom solution of Hash Message Authentication Code (HMAC).

Use Cases for Request Signing

If a COS object needs to be published to the public, you will usually need to set this object to public-read/private-write (everyone can read the object but only the ACL-specified account can write to it). After this, you can use the ACL policy together with the API request signature to authenticate requesters and control the permissions and validity period of operations.

> Note:
>
> The API request signing described in this document is already included if you're developing using an SDK. Only when you want to perform secondary development through the original API, follow the steps described in this document.

In the preceding scenario, various security measures can be applied to API requests.

1. Requester Identity Authentication Verify the requester's identity through their unique ID and key.
2. Transmitted data tampering prevention: Signs and verifies data to ensure the integrity of the transmitted content.
3. Signature theft prevention: Sets a validity period for the signature to avoid its theft and reuse.

Preparations

1. APPID, SecretId, and SecretKey:
   Obtain them on the API Key Management page in the CAM console.
2. Development language:
   Supported languages include but are not limited to Java, PHP, C#, C++, Node.js, and Python. Determine the HMAC-SHA1, SHA1, and UrlEncode functions based on the development language. The HMAC-SHA1 and SHA1 functions use UTF-8 encoded strings as inputs and hexadecimal lowercase strings as outputs. UrlEncode is based on UTF-8 encoding. For printable characters within the ASCII range, the special symbols listed in the following table should also be encoded.

Character	Decimal	Hexadecimal	Character	Decimal	Hexadecimal
(Space)	32	20	;	59	3B
!	33	21	<	60	3C
"	34	22	=	61	3D
#	35	23	>	62	3E
$	36	24	?	63	3F
%	37	25	@	64	40
&	38	26	[	91	5B
'	39	27	\	92	5C
(	40	28	]	93	5D
)	41	29	^	94	5E
*	42	2A	`	96	60
+	43	2B	{	123	7B
,	44	2C	|	124	7C
/	47	2F	}	125	7D
:	58	3A			

Signing Process

Generating KeyTime

1. Obtain the Unix timestamp StartTimestamp corresponding to the current time. The Unix timestamp

counts the total number of seconds from 00:00:00 on January 1, 1970 (UTC or GMT time, that is, 08:00:00 on January 1, 1970, Manila time) to the present.
2. Based on the preceding timestamp and the expected signature validity period, calculate the Unix timestamp EndTimestamp corresponding to the signature expiration time.
3. Concatenate the signature validity time, in StartTimestamp;EndTimestamp format, that is, KeyTime.
   Example: 1557902800;1557910000

Generating SignKey

Use HMAC-SHA1 with SecretKey as the key and KeyTime as the message to calculate the message digest (hash value), that is, SignKey.
Example: 36bcd76dbb8c9f066472fec403df8a34cab34c77

Generating UrlParamList and HttpParameters

1. Traverse HTTP request parameters to generate Map (mapping from keys to values) and KeyList (list of keys). Keys are converted to lowercase, and values are encoded using UrlEncode.
   For parameters without values, the values are regarded as null strings. For example, if the request path is /?acl , it is presumed to be /?acl= .
2. Sort keys in KeyList in lexicographic order.
3. Encode the keys in Map and KeyList using UrlEncode and convert them to lowercase.
4. Concatenate each key-value pair in Map based on the order in KeyList, in key1=value1&key2=value2&key3=value3 format, that is, HttpParameters.
5. Concatenate each item in KeyList based on the order in KeyList, in key1;key2;key3 format, that is, UrlParamList.
   Example:

- Example 1:
  Request path: /?prefix=example-folder%2F&delimiter=%2F&max-keys=10
  UrlParamList: delimiter;max-keys;prefix
  HttpParameters: delimiter=%2F&max-keys=10&prefix=example-folder%2F

  > Note:
  >
  > When requests are sent, the request parameters in the request path are also encoded using UrlEncode. Therefore, do not perform UrlEncode multiple times.

- Example 2:
  Request path: /exampleobject?acl
  UrlParamList: acl
  HttpParameters: acl=

Generating HeaderList and HttpHeaders

Generating HeaderList and HttpHeaders follows the same process as generating UrlParamList and HttpParameters, except that HTTP request parameters are replaced with HTTP request headers. HttpParameters become HttpHeaders, and UrlParamList becomes HeaderList.

Example:
Request header:

```
Date: Thu, 16 May 2019 03:15:06 GMT
Host: <BucketName-APPID>.<Endpoint>
x-cos-acl: private
x-cos-grant-read: uin="100000000011"
```

The calculated HeaderList is `date;host;x-cos-acl;x-cos-grant-read` , and HttpHeaders is `date=Thu%2C%2016%20May%202019%2003%3A15%3A06%20GMT&host=examplebucket-1250000000.cos.city.yfm4.fsphere.cn&x-cos-acl=private&x-cos-grant-read=uin%3D%22100000000011%22` .

Generating HttpString

Generate HttpString based on the HTTP method, HTTP request path, HttpParameters, and HttpHeaders, in `HttpMethod\nUriPathname\nHttpParameters\nHttpHeaders\n` format.
HttpMethod is converted to lowercase, such as `get` or `put` . UriPathname is the request path, such as `/` or `/exampleobject` . \n is a line break. If any string is empty, the beginning and last line breaks should be retained, such as `get\n/exampleobject\n\n\n` .

Generating StringToSign

Generate StringToSign based on KeyTime and HttpString, in `sha1\nKeyTime\nSHA1(HttpString)\n` format. SHA1(HttpString) is the message digest calculated by using SHA1 on HttpString.

Generating Signature

Use HMAC-SHA1 with SignKey as the key and StringToSign as the message to calculate the message digest, that is, Signature.

Generating Signature

Generate the signature based on SecretId, KeyTime, HeaderList, UrlParamList, and Signature, in the following format:

```
q-sign-algorithm=sha1
&q-ak=SecretId
&q-sign-time=KeyTime
&q-key-time=KeyTime
&q-header-list=HeaderList
```

```
&q-url-param-list=UrlParamList
&q-signature=Signature
```

> Note:
>
> Line breaks in the preceding format are for easier reading only and are not included in the actual format.

Using a Signature

Signed HTTP requests sent to COS through the RESTful API can pass the signature in the following ways:

1. Include the signature in the standard HTTP Authorization header, such as `Authorization: q-sign-algorithm=sha1&q-ak=...&q-sign-time=1557989753;1557996953&...&q-signature=...` .
2. Use the signature as an HTTP request parameter (note UrlEncode), such as `/exampleobject?q-sign-algorithm=sha1&q-ak=...&q-sign-time=1557989753%3B1557996953&...&q-signature=...` .

   > Note:
   >
   > In the preceding examples, some specific signature content is omitted in `...` .

Sample Code

Pseudocode

```
KeyTime = [Now];[Expires]
SignKey = HMAC-SHA1([SecretKey], KeyTime)
HttpString = [HttpMethod]\n[HttpURI]\n[HttpParameters]\n[HttpHeaders]\n
StringToSign = sha1\nKeyTime\nSHA1(HttpString)\n
Signature = HMAC-SHA1(SignKey, StringToSign)
```

Example Message Digest Algorithms

The following examples describe how to call HMAC-SHA1 in different languages.

PHP

```
$sha1HttpString = sha1('ExampleHttpString');

$signKey = hash_hmac('sha1', 'ExampleKeyTime', 'YourSecretKey');
```

Java

```
import org.apache.commons.codec.digest.DigestUtils;
import org.apache.commons.codec.digest.HmacUtils;

String sha1HttpString = DigestUtils.sha1Hex("ExampleHttpString");

String signKey = HmacUtils.hmacSha1Hex("YourSecretKey", "ExampleKeyTime");
```

Python

```
import hmac
import hashlib

sha1 = hashlib.sha1()
sha1_http_string = sha1.update('ExampleHttpString'.encode('utf-8')).hexdigest()

sign_key = hmac.new('YourSecretKey'.encode('utf-8'), 'ExampleKeyTime'.encode('utf-8'), hashlib.sha1).h
exdigest()
```

Node.Js

```
var crypto = require('crypto');

var sha1HttpString = crypto.createHash('sha1').update('ExampleHttpString').digest('hex');
var signKey = crypto.createHmac('sha1', 'YourSecretKey').update('ExampleKeyTime').digest('hex');
```

Go

```
import (
    "crypto/hmac"
    "crypto/sha1"
)

h := sha1.New()
h.Write([]byte("ExampleHttpString"))
sha1HttpString := h.Sum(nil)

var hashFunc = sha1.New
h = hmac.New(hashFunc, []byte("YourSecretKey"))
h.Write([]byte("ExampleKeyTime"))
signKey := h.Sum(nil)
```

Practical Case

Preparations

Log in to the CAM console and go to the API key management page to obtain the APPID, SecretId, and SecretKey. The following table describes an example.

APPID	SecretId	SecretKey
1250000000	AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	BQYIM75p8x0iWVFSIgqEKwFprpRSVHlz

Uploading an Object

Original Request

```
PUT /exampleobject(%E8%85%BE%E8%AE%AF%E4%BA%91) HTTP/1.1
Date: Thu, 16 May 2019 06:45:51 GMT
Host: <BucketName-APPID>.<Endpoint>
Content-Type: text/plain
Content-Length: 13
Content-MD5: mQ/fVh815F3k6TAUm8m0eg==
x-cos-acl: private
x-cos-grant-read: uin="100000000011"

ObjectContent
```

Intermediate Variables

- KeyTime = 1557989151;1557996351
- SignKey = eb2519b498b02ac213cb1f3d1a3d27a3b3c9bc5f
- UrlParamList = (empty string)
- HttpParameters = (empty string)
- HeaderList = content-length;content-md5;content-type;date;host;x-cos-acl;x-cos-grant-read
- HttpHeaders = content-length=13&content-md5=mQ%2FfVh815F3k6TAUm8m0eg%3D%3D&content-type=text%2Fplain&date=Thu%2C%2016%20May%202019%2006%3A45%3A51%20GMT&host=examplebucket-1250000000.cos.city.yfm4.fsphere.cn&x-cos-acl=private&x-cos-grant-read=uin%3D%22100000000011%22
- HttpString = put\n/exampleobject(Converge Cloud)\n\ncontent-length=13&content-md5=mQ%2FfVh815F3k6TAUm8m0eg%3D%3D&content-type=text%2Fplain&date=Thu%2C%2016%20May%202019%2006%3A45%3A51%20GMT&host=examplebucket-1250000000.cos.city.yfm4.fsphere.cn&x-cos-acl=private&x-cos-grant-read=uin%3D%22100000000011%22\n

- StringToSign = sha1\n1557989151;1557996351\n8b2751e77f43a0995d6e9eb9477f4b685cca4172\n
- Signature = 3b8851a11a569213c17ba8fa7dcf2abec6935172

(empty string) indicates a string of length 0, and \n indicates a line break.

Signed Request

```
PUT /exampleobject(%E8%85%BE%E8%AE%AF%E4%BA%91) HTTP/1.1
Date: Thu, 16 May 2019 06:45:51 GMT
Host: <BucketName-APPID>.<Endpoint>
Content-Type: text/plain
Content-Length: 13
Content-MD5: mQ/fVh815F3k6TAUm8m0eg==
x-cos-acl: private
x-cos-grant-read: uin="100000000011"
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=15
57989151;1557996351&q-key-time=1557989151;1557996351&q-header-list=content-length;content-
md5;content-type;date;host;x-cos-acl;x-cos-grant-read&q-url-param-list=&q-signature=3b8851a11a5
69213c17ba8fa7dcf2abec6935172

ObjectContent
```

Downloading an Object

Original Request

```
GET /exampleobject(%E8%85%BE%E8%AE%AF%E4%BA%91)?response-content-type=application%2Fo
ctet-stream&response-cache-control=max-age%3D600 HTTP/1.1
Date: Thu, 16 May 2019 06:55:53 GMT
Host: <BucketName-APPID>.<Endpoint>
```

Intermediate Variables

- KeyTime = 1557989753;1557996953
- SignKey = 937914bf490e9e8c189836aad2052e4feeb35eaf
- UrlParamList = response-cache-control;response-content-type
- HttpParameters = response-cache-control=max-age%3D600&response-content-type=application%2Foctet-stream
- HeaderList = date;host
- HttpHeaders = date=Thu%2C%2016%20May%202019%2006%3A55%3A53%20GMT&host=examplebucket-1250000000.cos.city.yfm4.fsphere.cn
- HttpString = get\n/exampleobject(Converge Cloud)\nresponse-cache-control=max-age%3D600&response-content-type=application%2Foctet-

stream\ndate=Thu%2C%2016%20May%202019%2006%3A55%3A53%20GMT&host=examplebucket-1250000000.cos.city.yfm4.fsphere.cn\n

- StringToSign = sha1\n1557989753;1557996953\n54ecfe22f59d3514fdc764b87a32d8133ea611e6\n
- Signature = 01681b8c9d798a678e43b685a9f1bba0f6c0e012

  \n indicates a line break.

Signed Request

```
GET /exampleobject(%E8%85%BE%E8%AE%AF%E4%BA%91)?response-content-type=application%2Foctet-stream&response-cache-control=max-age%3D600 HTTP/1.1
Date: Thu, 16 May 2019 06:55:53 GMT
Host: <BucketName-APPID>.<Endpoint>
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=1557989753;1557996953&q-key-time=1557989753;1557996953&q-header-list=date;host&q-url-param-list=response-cache-control;response-content-type&q-signature=01681b8c9d798a678e43b685a9f1bba0f6c0e012
```

Operation List

COS-related APIs and descriptions are as follows:

Request Signature

Request Signature

Service API

Operation Name	API	Operation Description
Querying the Bucket List	GET Service	Query the list of all buckets under a specified account.

Bucket API

Basic Operations

Operation Name	API	Operation Description
Creating a Bucket	PUT Bucket	Create a bucket under a specified account.
Querying the Object List	GET Bucket (List Objects)	Query some or all objects in a bucket.
Retrieving Buckets and Their Permissions	HEAD Bucket	Verify if the bucket exists and if there are permissions to access it.
Deleting a Bucket	DELETE Bucket	Delete empty buckets under a specified account.
Querying the Object Version	GET Bucket Object Versions	Query some or all objects in a bucket and their historical version information

Access Control List (ACL)

Operation Name	API	Operation Description
Setting an ACL for a Bucket	PUT Bucket acl	Set an ACL for a specified bucket.
Querying the ACL of a Bucket	GET Bucket acl	Query the ACL of a bucket.

Cross-Origin Resource Sharing (CORS)

Operation Name	API	Operation Description
Setting CORS Configuration	PUT Bucket cors	Set the cross-origin access permissions for a bucket.
Querying CORS Configuration	GET Bucket cors	Query the cross-origin access configuration of a bucket.
Deleting CORS Configuration	DELETE Bucket cors	Delete the cross-origin access configuration of a bucket.

Lifecycle

Operation Name	API	Operation Description
Setting the Bucket Lifecycle	PUT Bucket lifecycle	Set bucket lifecycle management configuration.
Querying the Bucket Lifecycle Configuration	GET Bucket lifecycle	Query bucket lifecycle management configuration.
Deleting the Bucket Lifecycle Configuration	DELETE Bucket lifecycle	Delete bucket lifecycle management configuration.

Bucket Policy

Operation Name	API	Operation Description
Setting a Bucket Policy	PUT Bucket policy	Set the permission policy for the specified bucket.
Querying a Bucket Policy	GET Bucket policy	Query the permission policy for the specified bucket.
Deleting a Bucket Policy	DELETE Bucket policy	Delete the permission policy for the specified bucket.

Hotlink Protection (Referer)

Operation Name	API	Operation Description
Setting a Bucket Referer	PUT Bucket referer	Set bucket Referer allowlist or blocklist.
Querying a Bucket Referer	GET Bucket referer	Query bucket Referer allowlist or blocklist.

Tag

Operation Name	API	Operation Description
Setting Bucket Tags	PUT Bucket tagging	Set tags for an existing bucket.
Querying Bucket Tags	GET Bucket tagging	Query existing tags for the specified bucket.
Deleting Bucket Tags	DELETE Bucket tagging	Delete specific tags for the bucket.

Static Website (website)

Operation Name	API	Operation Description
Settings a Static Website	PUT Bucket website	Configure a static website for the bucket.
Querying a Static Website	GET Bucket website	Query associated static website configuration information for the bucket.
Deleting a Static Website	DELETE Bucket website	Delete static website configuration from the bucket.

Inventory

Operation Name	API	Operation Description
Setting an Inventory Task	PUT Bucket inventory	Create an inventory task for a bucket.
Creating a One-time Inventory Task	POST Bucket inventory	Create a one-time inventory task for a bucket.
Querying an Inventory Task	GET Bucket inventory	Query the configuration of a specified inventory of a bucket.
Querying all Inventory Tasks	List Bucket Inventory Configurations	Query all inventory tasks of a bucket.
Deleting an Inventory Task	DELETE Bucket inventory	Delete a specified inventory task of a bucket.

Versioning

Operation Name	API	Operation Description
Setting Versioning	PUT Bucket versioning	Enable or suspend versioning functionality for the bucket.
Querying Versioning	GET Bucket versioning	Query the versioning information of a bucket.

In-cloud Replication (replication)

Operation Name	API	Operation Description
Setting In-Cloud Replication Rules	PUT Bucket replication	Configures in-cloud replication rules for a bucket with versioning enabled.
Querying In-Cloud Replication Rules	GET Bucket replication	Queries the in-cloud replication configuration of a bucket.
Deleting In-Cloud Replication Rules	DELETE Bucket replication	Deletes the in-cloud replication configuration of a bucket.

Cross-Cloud Replication (CrossCloudReplication)

Operation Name	API	Operation Description
Setting Cross-Cloud Replication Rules	PUT Bucket CrossCloudReplication	Creates, updates, and deletes cross-cloud replication rules for the source bucket.
Querying Cross-Cloud Replication Rules	GET Bucket CrossCloudReplication	Queries the cross-cloud replication rules configured for the source bucket.
Deleting Cross-Cloud Replication Rules	PUT Bucket CrossCloudReplication	Creates, updates, and deletes cross-cloud replication rules for the source bucket.

Logging

Operation Name	API	Operation Description
Setting Log Management	PUT Bucket logging	Enable logging for the source bucket and saves the access logs of the source bucket to a specified destination bucket.
Querying Log Management	GET Bucket logging	Obtain the logging configuration of the source bucket.

Bucket Encryption

Operation Name	API	Operation Description
Setting Bucket Encryption	PUT Bucket encryption	Set default encryption configuration for a specified bucket.
Querying Bucket Encryption	GET Bucket encryption	Query default encryption configuration for a specified bucket.
Deleting Bucket Encryption	DELETE Bucket encryption	Delete the default encryption configuration for a specified bucket.

Object Locking

Operation Name	API	Operation Description
Setting Object Lock	PUT Bucket ObjectLockConfiguration	Set object locking for an existing bucket.
Querying the Object Lock	GET Bucket ObjectLockConfiguration	Query the effective object locking configuration.
Querying the Object Locking Expiration Date	GET Object retention	Query the object locking expiration date.

Object API

Basic Operations

Operation Name	API	Operation Description
Uploading Objects in Simple Upload Mode	PUT Object	Upload an object to a bucket.
Setting Object Replication	PUT Object - Copy	Copy a file to the destination path.
Uploading Objects Through Forms	POST Object	Uploading objects by using form requests
Downloading an Object	GET Object	Download an object to a local directory.
Querying Metadata of Objects	HEAD Object	Query the metadata of an object.
Deleting an Object	DELETE Object	Delete a specified object from a bucket.
Deleting Multiple Objects	DELETE Multiple Objects	Delete objects from a bucket in batches.
Checking CORS Configuration	OPTIONS Object	Use a preflight request to confirm whether sending a real cross-domain request is allowed.
Recovering Archived Objects	POST Object restore	Restore an archived object for access.

Access Control

Operation Name	API	Operation Description

Operation Name	API	Operation Description
Setting an ACL for an Object	PUT Object acl	Set an ACL for a specific object in a bucket.
Querying the ACL of an Object	GET Object acl	Query the ACL of an object.

Object Tag

Operation Name	API	Operation Description
Setting Object Tags	PUT Object tagging	Set tags for uploaded objects.
Querying Object Tags	GET Object tagging	Query existing object tags for a specified object.
Deleting Object Tags	DELETE Object tagging	Delete existing object tags for a specified object.

Multipart Upload

Operation Name	API	Operation Description
Initializing Multipart Upload	Initiate Multipart Upload	Initialize a multipart upload task.
Uploading Parts	Upload Part	Upload a file in parts.
Copying Parts	Upload Part - Copy	Copy another object as a part.
Completing Multipart Upload	Complete Multipart Upload	Complete multipart upload of the entire file.
Terminating Multipart Upload	Abort Multipart Upload	Terminate a multipart upload operation and delete uploaded parts.
Querying Multipart Upload	List Multipart Uploads	Query information about ongoing multipart upload tasks.
Querying Uploaded Parts	List Parts	Query uploaded parts in a specified multipart upload operation.

Data Processing APIs

Image Processing

For details, see Image Processing.

Service APIs
Querying the Bucket List

Description

The GET Service API is used to query the list of all buckets under a requester's account.

Request

Request Example

```
GET / HTTP/1.1
Host: <Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Authorization: Auth String (See Request Signature for details.)
>
> Host: To query the list of buckets in a specified region

Request Header

Common Headers

This request uses common request headers. For details, see Common Request Headers.

Non-common Headers

This request has no special request header information.

Request Body

The request body of this request is empty.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Common Response Headers.

Special Response Headers

This request has no special response header information.

Response Body

A successful query returns application/xml data that includes the object information in the bucket.

```
<?xml version="1.0" encoding="UTF-8" ?>
<ListAllMyBucketsResult>
  <Owner>
    <ID>string</ID>
    <DisplayName>string</DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <Name>string</Name>
      <Location>string</Location>
      <CreateDate>string</CreateDate>
    </Bucket>
    <Bucket>
      <Name>string</Name>
      <Location>string</Location>
      <CreateDate>string</CreateDate>
    </Bucket>
  </Buckets>
</ListAllMyBucketsResult>
```

The data are described as follows:

Node Name (Keyword)	Parent Node	Description	Type	Required
ListAllMyBucketsResult	No	Describes all the information of this response	Container	Yes

Content of the ListAllMyBucketsResult node of the container type:

Node Name (Keyword)	Parent Node	Description	Type	Required
Owner	ListAllMyBucketsResult	Describes bucket owner information	Container	Yes
Buckets	ListAllMyBucketsResult	Describes all bucket list information of this response	Container	Yes

Content of Owner:

Node Name (Keyword)	Parent Node	Description	Type	Required
ID	ListAllMyBucketsResult.Owner	ID of the bucket owner.	string	Yes
DisplayName	ListAllMyBucketsResult.Owner	Name of the bucket owner.	string	Yes

Content of the Buckets node of the container type:

Node Name (Keyword)	Parent Node	Description	Type	Required
Bucket	ListAllMyBucketsResult.Buckets	Single Bucket Information	Container	Yes

Content of the Bucket node of the container type:

Node Name (Keyword)	Parent Node	Description	Type	Required
Name	ListAllMyBucketsResult.Buckets.Bucket	Bucket	string	Yes
Location	ListAllMyBucketsResult.Buckets.Bucket	Region where the bucket is located.	string	Yes
CreateDate	ListAllMyBucketsResult.Buckets.Bucket	Bucket creation time in ISO 8601 format. Example: 2019-05-24T10:56:40Z.	string	Yes

Error Code

Error Code	Description	HTTP Status Code
InvalidBucketName	The bucket name is invalid.	400 Bad Request
SignatureDoesNotMatch	If this error code is returned, it means the provided signature is incorrect.	403 Forbidden

Error Code	Description	HTTP Status Code
NoSuchBucket	If this error code is returned, it means there is no such bucket.	404 Not Found

Practical Case

Request

```
GET / HTTP/1.1
Host: cos.city.yfm4.fsphere.cn
Date: Fri, 24 May 2019 11:59:51 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKID8A0fBVtYFrNm02oY1g1JQQF0c3JO****&q-sign-time=1558699191;1558706391&q-key-time=1558699191;1558706391&q-header-list=date;host&q-url-param-list=&q-signature=c3f55f4ce2800fb343cf85ff536a9185a0c1****
Connection: close
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 495
Connection: close
Date: Fri, 24 May 2019 11:59:51 GMT
x-cos-request-id: NWNlN2RjYjdfZjhjODBiMDlfOWNlNF9hYzc2****

<ListAllMyBucketsResult>
  <Owner>
    <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
    <DisplayName>100000000001</DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <Name>examplebucket1-1250000000</Name>
      <Location>city</Location>
      <CreationDate>2019-05-24T11:49:50Z</CreationDate>
    </Bucket>
    <Bucket>
      <Name>examplebucket2-1250000000</Name>
      <Location>city</Location>
      <CreationDate>2019-05-24T11:51:50Z</CreationDate>
    </Bucket>
  </Buckets>
```

```
</ListAllMyBucketsResult>
```

Bucket APIs
Basic Operations
Creating a Bucket

Description

This API is used to create a bucket under a specified account. A signature needs to be carried in Authorization and anonymous calls are not supported. By default, the bucket creator is the bucket owner.

Notes

If no access permission is specified for a bucket when it is created, private (private read/write) will be used by default.

Request

Syntax example:

```
PUT / HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Line

```
PUT / HTTP/1.1
```

This API accepts PUT requests.

Request Header

Common headers:
This request is implemented by using common request headers. For details on common request headers, see the Common Request Headers section.

Non-common headers:

This request is implemented by using the x-cos-acl header in PUT requests to set the bucket access permissions. Currently, bucket access permissions are divided into three types: public-read-write, public-read, and private. If the permission is not set, private is configured by default. The read, write, or read-write permissions can also be granted individually to users. Details are as follows:

> For more information about ACL requests, see Put Bucket ACL.

Name	Description	Type	Required
x-cos-acl	Defines the acl attribute of an object. Valid values are private, public-read, and public-read-write, and the default value is private.	String	No
x-cos-grant-read	Grants read access to the grantee. Format: x-cos-grant-read: id=" ",id=" "; Grants access permissions to root account only, id="qcs::cam::uin/ :uin/"	String	No
x-cos-grant-write	Grants write permission to the grantee. Format: x-cos-grant-write: id=" ",id=" "; Grants access permissions to root account only, id="qcs::cam::uin/ :uin/"	String	No
x-cos-grant-full-control	Grants read/write permission to the grantee. Format: x-cos-grant-full-control: id=" ",id=" "; Grants access permissions to root account only, id="qcs::cam::uin/ :uin/"	String	No

Request Body

The request body of this request is empty.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Public Response Header.

Special Response Headers

There are no special response headers for this response operation.

Response Body

The response body is empty.

Error Analysis

The following describes some special but common errors that may occur with this request:

Error Code	HTTP Status Code	Description
BucketAlreadyExists	409 Conflict	The specified bucket already exists and is owned by the current account.
InvalidBucketName	400 Bad Request	The bucket name is invalid. For more information, see message description.
InvalidRequest	400 Bad Request	The bucket name is invalid. For more information, see message description.

If the ACL set for the bucket is incorrect, bucket creation will fail and the error message Failed to set access control authority for the bucket will be returned. For the specific error cause, see Put Bucket ACL according to the returned error code.
For more information on COS error codes or the complete list of product errors, please refer to the Error Codes document.

Practical Case

Request

```
PUT / HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Thu, 12 Jan 2016 19:12:22 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84708728;32557604728&q-key-time=1484708728;32557604728&q-header-list=host&q-url-param-list
=&q-signature=b394a86624cbcc705b11bc6fc505843c5e2dd9c9
```

Response

```
HTTP /1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Connection: keep-alive
Date: Thu, 12 Jan 2016 19:12:22 GMT
x-cos-request-id: NTg3ZWRiODJfOWIxZjRlXzZmNDBfMTUz
```

Querying the Object List

Description

The GET Bucket request is equivalent to the List Objects request and can be used to list some or all objects in a bucket. To call this API, you need to have permission to read the bucket.

Request

Request Example

```
GET / HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Header

Common Headers

This request uses common request headers. For details, see Common Request Headers.

Non-common Headers

This request has no special request header information.

Request Parameters

Name	Type	Description	Required
prefix	string	Prefix matching, used to specify the file prefix address to be returned.	No
delimiter	string	The delimiter is a character. If there is prefix is specified, the same paths from prefix to delimiter are grouped into one category and defined as Common Prefixe and all Common Prefixes are listed. If no prefix is specified, it starts from the beginning of the path.	No
encoding-type	string	Specifies Encoding type of the returned value. Valid value: url.	No

Name	Type	Description	Required
marker	string	By default, entries are listed in UTF-8 binary order, starting from the marker.	No
max-keys	string	The maximum number (up to 1,000) of keys returned in the response. Defaults to 1000.	No

Request Body

The request does not have a request body.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Common Response Headers.

Special Response Headers

There are no special response headers for this response operation.

Response Body

A successful query returns application/xml data that includes the object information in the bucket.

```
<?xml version='1.0' encoding='utf-8' ?>
<ListBucketResult>
  <Name>examplebucket-1250000000</Name>
  <Encoding-Type>url</Encoding-Type>
  <Prefix>ela</Prefix>
  <Marker/>
  <MaxKeys>1000</MaxKeys>
  <Delimiter>/</Delimiter>
  <IsTruncated>false</IsTruncated>
  <NextMarker>exampleobject.txt</NextMarker>
  <Contents>
    <Key>photo</Key>
    <LastModified>2017-06-23T12:33:26.000Z</LastModified>
    <ETag>\"79f2a852fac7e826c9f4dbe037f8a63b\"</ETag>
    <Size>10485760</Size>
    <Owner>
      <ID>1250000000</ID>
    </Owner>
```

```
      <StorageClass>STANDARD</StorageClass>
    </Contents>
    <CommonPrefixes>
      <Prefix>example</Prefix>
    </CommonPrefixes>
</ListBucketResult>
```

The data are described as follows:

Node Name (Keyword)	Parent Node	Description	Type
ListBucketResult	No	Stores the result of the GET Bucket request.	Container

Content of ListBucketResult:

Node Name (Keyword)	Parent Node	Description	Type
Name	ListBucketResult	Bucket information	string
Encoding-Type	ListBucketResult	Encoding Format	string
Prefix	ListBucketResult	Prefix matching is used to specify the prefix address of the file returned by the response request.	string
Marker	ListBucketResult	By default, entries are listed in UTF-8 binary order, starting from the marker.	string
MaxKeys	ListBucketResult	Maximum number of keys returned in a single response request.	string
Delimiter	ListBucketResult	The delimiter is a character. If there is prefix is specified, the same paths from prefix to delimiter are grouped into one category and defined as Common Prefixe and all Common Prefixes are listed. If no prefix is specified, it starts from the beginning of the path.	string
IsTruncated	ListBucketResult	Indicates whether the returned list is truncated. Valid Boolean value: true, false.	boolean
NextMarker	ListBucketResult	If the returned entries are truncated, NextMarker is the starting point of the next entry.	string
Contents	ListBucketResult	Metadata Information	Container

Node Name (Keyword)	Parent Node	Description	Type
CommonPrefixes	ListBucketResult	The same paths from Prefix to delimiter are grouped into one category and defined as Common Prefix.	Container

Content of Contents:

Node Name (Keyword)	Parent Node	Description	Type
Key	ListBucketResult.Contents	Object Key	string
LastModified	ListBucketResult.Contents	Time the object was last modified.	string
ETag	ListBucketResult.Contents	The MD-5 algorithm checksum of the file.	string
Size	ListBucketResult.Contents	Object size, in bytes	string
Owner	ListBucketResult.Contents	Information of the object owner	Container
StorageClass	ListBucketResult.Contents	Object storage class. Enumerated values: STANDARD, STANDARD_IA, ARCHIVE.	string

Content of Owner:

Node Name (Keyword)	Parent Node	Description	Type
ID	ListBucketResult.Contents.Owner	APPID of the Bucket	string

Content of CommonPrefixes:

Node Name (Keyword)	Parent Node	Description	Type
Prefix	ListBucketResult.CommonPrefixes	A single common prefix string	string

Error Code

This request returns common error responses and error codes. For more information, see Error Codes.

Practical Case

Request

```
GET / HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Wed, 18 Oct 2014 22:32:00 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84213451;32557109451&q-key-time=1484213451;32557109451&q-header-list=host&q-url-param-list
=&q-signature=0336a1fc8350c74b6c081d4dff8e7a2db9007dc
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 1132
Connection: keep-alive
Vary: Accept-Encoding
Date: Wed, 18 Oct 2014 22:32:00 GMT
x-cos-request-id: NTg3NzRjY2VfYmRjMzVfMTc5M182MmIyNg==

<?xml version='1.0' encoding='utf-8' ?>
<ListBucketResult>
    <Name>examplebucket-1250000000</Name>
    <Encoding-Type>url</Encoding-Type>
    <Prefix>ela</Prefix>
    <Marker/>
    <MaxKeys>1000</MaxKeys>
    <Delimiter>/</Delimiter>
    <IsTruncated>false</IsTruncated>
    <NextMarker>exampleobject.txt</NextMarker>
    <Contents>
        <Key>photo</Key>
        <LastModified>2017-06-23T12:33:26.000Z</LastModified>
        <ETag>\"79f2a852fac7e826c9f4dbe037f8a63b\"</ETag>
        <Size>10485760</Size>
        <Owner>
            <ID>1250000001</ID>
        </Owner>
    <StorageClass>STANDARD</StorageClass>
    </Contents>
    <Contents>
        <Key>picture</Key>
        <LastModified>2017-06-23T12:33:26.000Z</LastModified>
        <ETag>\"3f9a5dbff88b25b769fa6304902b5d9d\"</ETag>
```

```
        <Size>10485760</Size>
        <Owner>
            <ID>1250000002</ID>
        </Owner>
    <StorageClass>STANDARD</StorageClass>
    </Contents>
    <Contents>
        <Key>file</Key>
        <LastModified>2017-06-23T12:33:26.000Z</LastModified>
        <ETag>\"39bfb88c11c65ed6424d2e1cd4db1826\"</ETag>
        <Size>10485760</Size>
        <Owner>
            <ID>1250000003</ID>
        </Owner>
    <StorageClass>STANDARD</StorageClass>
    </Contents>
    <Contents>
        <Key>world</Key>
        <LastModified>2017-06-23T12:33:26.000Z</LastModified>
        <ETag>\"fb31459ad10289ff49327fd91a3e1f6a\"</ETag>
        <Size>4</Size>
        <Owner>
            <ID>1250000004</ID>
        </Owner>
        <StorageClass>STANDARD</StorageClass>
    </Contents>
</ListBucketResult>
```

Retrieving Buckets and Their Permissions

Description

HEAD Bucket is used to verify whether a bucket exists and whether you have permission to access it. The access permission for HEAD is identical to read permission. Possible results are as follows:

- If the bucket exists and you have permission to read it, HTTP status code 200 will be returned.
- If you do not have permission to read the bucket, HTTP status code 403 will be returned.
- If the bucket does not exist, HTTP status code 404 will be returned.

> Note:
>
> Currently, we do not have a public API for getting Bucket attributes (returning information such as acl).

Request

Request Example

```
HEAD / HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Header

Common Headers

This request uses common request headers. For details, see Common Request Headers.

Non-common Headers

This request has no special request header information.

Request Body

The request does not have a request body.

Response

Response Headers

Common Response Headers

This response contains common response headers. For details, see Common Response Headers.

Special Response Headers

There are no special response headers for this response operation.

Response Body

The response body is empty.

Practical Case

Request

```
HEAD / HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Thu, 27 Oct 2015 20:32:00 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84640517;32557536517&q-key-time=1484640517;32557536517&q-header-list=host&q-url-param-list
=&q-signature=7bedc2f84a0a3d29df85fe727d0c1e388c410376
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Date: Thu, 27 Oct 2015 20:32:00 GMT
x-cos-request-id: NTg3ZGQxNDNfNDUyMDRlXzUyOWNfMjY5
```

Deleting a Bucket

Description

Before deleting a bucket, make sure that all the data and incomplete multipart uploads in the bucket have been deleted; otherwise, the bucket cannot be deleted.

Request

Syntax example:

```
DELETE / HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Line

```
DELETE / HTTP/1.1
```

This API accepts DELETE requests.

Request Header

Common Headers

This request uses common request headers. For details, see Common Request Headers.

Non-common Headers

This request has no special request header information.

Request Body

The request body of this request is empty.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Public Response Header.

Special Response Headers

There are no special response headers for this response operation.

Response Body

The response body is empty.

Error Analysis

The following describes some special but common errors that may occur with this request:

Error Code	Description	HTTP Status Code
BucketNotEmpty	409 Conflict	The bucket is not empty and cannot be deleted.
AccessDenied	403 Forbidden	Deleting a bucket also requires a signature. If this error code is returned, it means that you do not have permission to access the bucket you want to delete.
NoSuchBucket	404 Not Found	If this error code is returned, it means the bucket you want to delete is not found.

For more information on COS error codes or the complete list of product errors, please refer to the Error Codes document.

Practical Case

Request

```
DELETE / HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Wed, 23 Oct 2016 21:32:00 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84708950;32557604950&q-key-time=1484708950;32557604950&q-header-list=host&q-url-param-list
=&q-signature=2b27b72ad2540ff2dde341dc7579a66ee8cb2afc
```

Response

```
HTTP/1.1 204 OK
```

```
Content-Type: application/xml
Content-Length: 0
Connection: keep-alive
Date: Wed, 23 Oct 2016 21:32:00 GMT
x-cos-request-id: NTg3ZWRjNjBfOTgxZjRlXzZhYjlfMTg0
```

Querying the Object Version

Description

The GET Bucket Object versions API is used to retrieve all objects in a bucket and their historical version information. You can filter out part of the objects in the bucket and their historical version information by specifying parameters. The requester of this API needs the read permission for the bucket.

Request

Request Example

```
GET /?versions HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (For details, see Request Signature).

Request Parameters

Name	Description	Type	Required
prefix	Prefix used to match object keys, ensuring that a response only includes object keys with the specified prefix.	string	No
delimiter	One-character separator used for grouping object keys. Same content from the prefix or from the start (if no prefix is specified) to the first delimiter in all object keys is regarded as a prefix node under CommonPrefixes. The grouped object keys do not appear in the subsequent object list. For specific scenarios and usage, see the examples below.	string	No
encoding-type	Encoding method of the return value. An optional value is url, indicating that the returned object key is a URL-encoded (percent-encoded) value. For example, "Converge Cloud" is encoded as %E8%85%BE%E8%AE%AF%E4%BA%91 .	string	No
key-marker	Starting object key marker, starting from which (excluding), object version entries are returned in UTF-8 lexicographic order.	string	No

Name	Description	Type	Required
version-id-marker	Starting version ID marker, starting from which (excluding), object version entries are returned.	string	No
max-keys	The maximum number (up to 1,000) of keys returned in the response. Defaults to 1000.	integer	No

Request Header

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

This API has no request body.

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

The query is successful, and application/xml data is returned, including the object version information in the bucket. For response bodies in different scenarios, see the following examples.

```
<ListVersionsResult>
  <Name>string</Name>
  <Prefix>string</Prefix>
  <KeyMarker>string</KeyMarker>
  <VersionIdMarker>string</VersionIdMarker>
  <MaxKeys>integer</MaxKeys>
  <IsTruncated>boolean</IsTruncated>
  <Delimiter>string</Delimiter>
  <CommonPrefixes>
    <Prefix>string</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>string</Prefix>
  </CommonPrefixes>
  <Version>
    <Key>string</Key>
    <VersionId>string</VersionId>
    <IsLatest>boolean</IsLatest>
```

```
        <LastModified>date</LastModified>
        <ETag>string</ETag>
        <Size>integer</Size>
        <StorageClass>Enum</StorageClass>
        <Owner>
          <ID>string</ID>
          <DisplayName>string</DisplayName>
        </Owner>
      </Version>
      <DeleteMarker>
        <Key>string</Key>
        <VersionId>string</VersionId>
        <IsLatest>boolean</IsLatest>
        <LastModified>date</LastModified>
        <Owner>
          <ID>string</ID>
          <DisplayName>string</DisplayName>
        </Owner>
      </DeleteMarker>
      <DeleteMarker>
        <Key>string</Key>
        <VersionId>string</VersionId>
        <IsLatest>boolean</IsLatest>
        <LastModified>date</LastModified>
        <Owner>
          <ID>string</ID>
          <DisplayName>string</DisplayName>
        </Owner>
      </DeleteMarker>
      <Version>
        <Key>string</Key>
        <VersionId>string</VersionId>
        <IsLatest>boolean</IsLatest>
        <LastModified>date</LastModified>
        <ETag>string</ETag>
        <Size>integer</Size>
        <StorageClass>Enum</StorageClass>
        <Owner>
          <ID>string</ID>
          <DisplayName>string</DisplayName>
        </Owner>
      </Version>
    </ListVersionsResult>
```

Specific node descriptions are as follows:

Node Name	Parent	Description	Type

(Keyword)	Node		
ListVersionsResult	No	Stores the result of GET Bucket Object versions.	Container

Content of ListVersionsResult:

Node Name (Keyword)	Parent Node	Description	Type
Name	ListVersionsResult	Name of the bucket, in <BucketName-APPID> format, for example, examplebucket-1250000000 .	string
EncodingType	ListVersionsResult	Encoding format, corresponding to the encoding-type parameter in the request. This node is returned only when the encoding-type parameter is carried in the request.	string
Prefix	ListVersionsResult	Prefix used to match object keys, corresponding to the prefix parameter in the request.	string
KeyMarker	ListVersionsResult	Starting object key marker, starting from which (excluding), object version entries are returned in UTF-8 lexicographic order. It corresponds to the key-marker parameter in the request.	string
VersionIdMarker	ListVersionsResult	Starting version ID marker, starting from which (excluding), object version entries are returned. It corresponds to the version-id-marker parameter in the request.	string
MaxKeys	ListVersionsResult	Maximum number of entries returned in a single response. It corresponds to the max-keys parameter in the request.	integer
Delimiter	ListVersionsResult	Separator. It corresponds to the delimiter parameter in the request, and this node is returned only if the delimiter parameter is carried in the request.	string
IsTruncated	ListVersionsResult	Whether the response entries are truncated. It is set to a boolean value, for example, true or false.	boolean

Node Name (Keyword)	Parent Node	Description	Type
NextKeyMarker	ListVersionsResult	This node is returned only if the response entries are truncated (IsTruncated is true). The value of this node is the last object key in the response entries. To continue requesting subsequent entries, use the value of this node as the key-marker parameter for the next request.	string
NextVersionIdMarker	ListVersionsResult	This node is returned only if the response entries are truncated (IsTruncated is true). The value of this node is the version ID of the last object in the response entries. To continue requesting subsequent entries, use the value of this node as the version-id-marker parameter for the next request.	string
CommonPrefixes	ListVersionsResult	The identical paths between the prefix (or, if no prefix is specified, the beginning of the string) and the first delimiter are grouped and defined as a common prefix. This node will be returned only if the delimiter parameter is specified in the request.	Container
Version	ListVersionsResult	Object version entry.	Container
DeleteMarker	ListVersionsResult	Object delete marker entry.	Container

Content of the ListVersionsResult.CommonPrefixes node of the Container type:

Node Name (Keyword)	Parent Node	Description	Type
Prefix	ListVersionsResult.CommonPrefixes	A single common prefix.	string

Content of the ListVersionsResult.Version node of the Container type:

Node Name (Keyword)	Parent Node	Description	Type
Key	ListVersionsResult.Version	Object Key	string
VersionId	ListVersionsResult.Version	Version ID of an object.	string

Node Name (Keyword)	Parent Node	Description	Type
IsLatest	ListVersionsResult.Version	Whether the current version is the latest version of the object.	boolean
LastModified	ListVersionsResult.Version	Last modification time of the current version, in ISO 8601 format, for example, 2019-05-24T10:56:40Z.	date
ETag	ListVersionsResult.Version	Entity tag (ETag) of the object, which is an information tag identifying the content of an object when it is created. It can be used to check whether the content of the object is changed, for example, "8e0b617ca298a564c3331da28dcb50df". This header might not return the MD5 value of the object. It varies depending on the object upload and encryption methods.	string
Size	ListVersionsResult.Version	Object size, in bytes.	integer
StorageClass	ListVersionsResult.Version	Storage type of an object. For enumerated values, see [Storage Type], for example, STANDARD_IA and ARCHIVE.	Enum
Owner	ListVersionsResult.Version	Information about the object owner.	Container

Content of the ListVersionsResult.Version.Owner node of the Container type:

Node Name (Keyword)	Parent Node	Description	Type
ID	ListVersionsResult.Version.Owner	App ID of the object owner.	string
DisplayName	ListVersionsResult.Version.Owner	Name of the object owner.	string

Content of the ListVersionsResult.DeleteMarker node of the Container type:

Node Name (Keyword)	Parent Node	Description	Type
Key	ListVersionsResult.DeleteMarker	Object Key	string
VersionId	ListVersionsResult.DeleteMarker	Version ID of the object delete marker.	string

Node Name (Keyword)	Parent Node	Description	Type
IsLatest	ListVersionsResult.DeleteMarker	Whether the object delete marker is the latest version.	boolean
LastModified	ListVersionsResult.DeleteMarker	Deletion time of the delete marker, in ISO8601 format, such as 2019-05-24T10:56:40Z.	date
Owner	ListVersionsResult.DeleteMarker	Information about the object owner.	Container

Content of the ListVersionsResult.DeleteMarker.Owner node of the Container type

Node Name (Keyword)	Parent Node	Description	Type
ID	ListVersionsResult.DeleteMarker.Owner	App ID of the object owner.	string
DisplayName	ListVersionsResult.DeleteMarker.Owner	Name of the object owner.	string

Error Code

This API does not have any special error information. For all error information, see Error Code.

Practical Case

Example 1: Versioning Is Not Enabled

Request

```
GET /?versions HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Thu, 15 Aug 2019 12:03:09 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKID8A0fBVtYFrNm02oY1g1JQQF0c3JO****&q-sign-time=1565870589;1565877789&q-key-time=1565870589;1565877789&q-header-list=date;host&q-url-param-list=versions&q-signature=1d8fcb8522df7be9fa52d94cd79462f92eb3****
Connection: close
```

Response

```
HTTP/1.1 200 OK
```

```
Content-Type: application/xml
Content-Length: 1262
Connection: close
Date: Thu, 15 Aug 2019 12:03:10 GMT
x-cos-request-id: NWQ1NTQ5ZmVfYjliYjBiMDlfMmFhNGZfY2Jm****

<ListVersionsResult>
    <Name>examplebucket-1250000000</Name>
    <Prefix/>
    <KeyMarker/>
    <VersionIdMarker/>
    <MaxKeys>1000</MaxKeys>
    <IsTruncated>false</IsTruncated>
    <Version>
        <Key>example-object-1.jpg</Key>
        <VersionId/>
        <IsLatest>true</IsLatest>
        <LastModified>2019-08-15T12:03:10.000Z</LastModified>
        <ETag>&amp;quot;0f0cd12c48979d1bf3f95255a36cb861&amp;quot;</ETag>
        <Size>20</Size>
        <StorageClass>STANDARD</StorageClass>
        <Owner>
            <ID>1250000000</ID>
            <DisplayName>1250000000</DisplayName>
        </Owner>
    </Version>
    <Version>
        <Key>example-object-2.jpg</Key>
        <VersionId/>
        <IsLatest>true</IsLatest>
        <LastModified>2019-08-15T12:03:09.000Z</LastModified>
        <ETag>&amp;quot;51370fc64b79d0d3c7c609635be1c41f&amp;quot;</ETag>
        <Size>20</Size>
        <StorageClass>STANDARD</StorageClass>
        <Owner>
            <ID>1250000000</ID>
            <DisplayName>1250000000</DisplayName>
        </Owner>
    </Version>
    <Version>
        <Key>example-object-3.jpg</Key>
        <VersionId/>
        <IsLatest>true</IsLatest>
        <LastModified>2019-08-15T12:03:08.000Z</LastModified>
        <ETag>&amp;quot;b2f1d893c5fde000ee8ea6eca18ed81f&amp;quot;</ETag>
        <Size>20</Size>
        <StorageClass>STANDARD</StorageClass>
```

```
        <Owner>
          <ID>1250000000</ID>
          <DisplayName>1250000000</DisplayName>
        </Owner>
      </Version>
    </ListVersionsResult>
```

Example 2: Versioning Is Enabled

Request

```
GET /?versions HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Thu, 15 Aug 2019 12:03:41 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKID8A0fBVtYFrNm02oY1g1JQQF0c3JO****&q-sign-tim
e=1565870621;1565877821&q-key-time=1565870621;1565877821&q-header-list=date;host&q-url-pa
ram-list=versions&q-signature=36400914186e87b3e88cc8049a79da5e3d79****
Connection: close
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 3477
Connection: close
Date: Thu, 15 Aug 2019 12:03:41 GMT
x-cos-request-id: NWQ1NTRhMWRfODhjMjJhMDlfMWNkOF8xZTRm****

<ListVersionsResult>
    <Name>examplebucket-1250000000</Name>
    <Prefix/>
    <KeyMarker/>
    <VersionIdMarker/>
    <MaxKeys>1000</MaxKeys>
    <IsTruncated>false</IsTruncated>
    <Version>
        <Key>example-object-1.jpg</Key>
        <VersionId>null</VersionId>
        <IsLatest>true</IsLatest>
        <LastModified>2019-08-15T12:03:10.000Z</LastModified>
        <ETag>&amp;quot;0f0cd12c48979d1bf3f95255a36cb861&amp;quot;</ETag>
        <Size>20</Size>
        <StorageClass>STANDARD</StorageClass>
        <Owner>
            <ID>1250000000</ID>
            <DisplayName>1250000000</DisplayName>
```

```
    </Owner>
</Version>
<DeleteMarker>
    <Key>example-object-2.jpg</Key>
    <VersionId>MTg0NDUxNzgyMDMwODg0NzI0Njc</VersionId>
    <IsLatest>true</IsLatest>
    <LastModified>2019-08-15T12:03:41.000Z</LastModified>
    <Owner>
        <ID>1250000000</ID>
        <DisplayName>1250000000</DisplayName>
    </Owner>
</DeleteMarker>
<Version>
    <Key>example-object-2.jpg</Key>
    <VersionId>null</VersionId>
    <IsLatest>false</IsLatest>
    <LastModified>2019-08-15T12:03:09.000Z</LastModified>
    <ETag>&amp;quot;51370fc64b79d0d3c7c609635be1c41f&amp;quot;</ETag>
    <Size>20</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
        <ID>1250000000</ID>
        <DisplayName>1250000000</DisplayName>
    </Owner>
</Version>
<DeleteMarker>
    <Key>example-object-3.jpg</Key>
    <VersionId>MTg0NDUxNzgyMDMwODg0NzA1NzM</VersionId>
    <IsLatest>true</IsLatest>
    <LastModified>2019-08-15T12:03:41.000Z</LastModified>
    <Owner>
        <ID>1250000000</ID>
        <DisplayName>1250000000</DisplayName>
    </Owner>
</DeleteMarker>
<Version>
    <Key>example-object-3.jpg</Key>
    <VersionId>MTg0NDUxNzgyMDMwODk5NTUxNzU</VersionId>
    <IsLatest>false</IsLatest>
    <LastModified>2019-08-15T12:03:39.000Z</LastModified>
    <ETag>&amp;quot;e5c7403f4ac3ace73477eb8b1fd183f7&amp;quot;</ETag>
    <Size>30</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
        <ID>1250000000</ID>
        <DisplayName>1250000000</DisplayName>
    </Owner>
```

```
</Version>
<Version>
  <Key>example-object-3.jpg</Key>
  <VersionId>null</VersionId>
  <IsLatest>false</IsLatest>
  <LastModified>2019-08-15T12:03:08.000Z</LastModified>
  <ETag>&amp;quot;b2f1d893c5fde000ee8ea6eca18ed81f&amp;quot;</ETag>
  <Size>20</Size>
  <StorageClass>STANDARD</StorageClass>
  <Owner>
    <ID>1250000000</ID>
    <DisplayName>1250000000</DisplayName>
  </Owner>
</Version>
<Version>
  <Key>example-object-4.jpg</Key>
  <VersionId>MTg0NDUxNzgyMDMwODg4OTI0MDc</VersionId>
  <IsLatest>true</IsLatest>
  <LastModified>2019-08-15T12:03:41.000Z</LastModified>
  <ETag>&amp;quot;c5c4d52f90ec328890953bbe4ae08230&amp;quot;</ETag>
  <Size>30</Size>
  <StorageClass>STANDARD</StorageClass>
  <Owner>
    <ID>1250000000</ID>
    <DisplayName>1250000000</DisplayName>
  </Owner>
</Version>
<Version>
  <Key>example-object-4.jpg</Key>
  <VersionId>MTg0NDUxNzgyMDMwODkyMTg2NjI</VersionId>
  <IsLatest>false</IsLatest>
  <LastModified>2019-08-15T12:03:40.000Z</LastModified>
  <ETag>&amp;quot;e9ec8bcb980d2e4d8526c346eb3b2585&amp;quot;</ETag>
  <Size>20</Size>
  <StorageClass>STANDARD</StorageClass>
  <Owner>
    <ID>1250000000</ID>
    <DisplayName>1250000000</DisplayName>
  </Owner>
</Version>
<Version>
  <Key>example-object-5.jpg</Key>
  <VersionId>MTg0NDUxNzgyMDMwODk4NTkzMzM</VersionId>
  <IsLatest>true</IsLatest>
  <LastModified>2019-08-15T12:03:39.000Z</LastModified>
  <ETag>&amp;quot;201669a14bdf051d8a9d6f9828d3f4c4&amp;quot;</ETag>
  <Size>20</Size>
```

```
        <StorageClass>STANDARD</StorageClass>
        <Owner>
          <ID>1250000000</ID>
          <DisplayName>1250000000</DisplayName>
        </Owner>
    </Version>
</ListVersionsResult>
```

Example 3: The delimiter Parameter Is Carried (Listing Objects and Subdirectories Under the Root Directory)

Request

```
GET /?versions&delimiter=%2F HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Fri, 16 Aug 2019 10:45:53 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKID8A0fBVtYFrNm02oY1g1JQQF0c3JO****&q-sign-tim
e=1565952353;1565959553&q-key-time=1565952353;1565959553&q-header-list=date;host&q-url-pa
ram-list=delimiter;versions&q-signature=c3130139bcac870247d1a070dbc8ee1c7ad5****
Connection: close
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 2529
Connection: close
Date: Fri, 16 Aug 2019 10:45:53 GMT
x-cos-request-id: NWQ1Njg5NjFfNjFiMDJhMDlfMWE5ZV8yMDY4****

<ListVersionsResult>
    <Name>examplebucket-1250000000</Name>
    <Prefix/>
    <KeyMarker/>
    <VersionIdMarker/>
    <MaxKeys>1000</MaxKeys>
    <IsTruncated>false</IsTruncated>
    <Delimiter>/</Delimiter>
    <CommonPrefixes>
        <Prefix>example-folder-1/</Prefix>
    </CommonPrefixes>
    <CommonPrefixes>
        <Prefix>example-folder-2/</Prefix>
    </CommonPrefixes>
    <Version>
        <Key>example-object-1.jpg</Key>
```

```
    <VersionId>MTg0NDUxNzgxMjEzNTU3NTk1Mjg</VersionId>
    <IsLatest>true</IsLatest>
    <LastModified>2019-08-16T10:45:53.000Z</LastModified>
    <ETag>&amp;quot;5d1143df07a17b23320d0da161e2819e&amp;quot;</ETag>
    <Size>30</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>1250000000</ID>
      <DisplayName>1250000000</DisplayName>
    </Owner>
  </Version>
  <DeleteMarker>
    <Key>example-object-1.jpg</Key>
    <VersionId>MTg0NDUxNzgxMjEzNjE1OTcxMzM</VersionId>
    <IsLatest>false</IsLatest>
    <LastModified>2019-08-16T10:45:47.000Z</LastModified>
    <Owner>
      <ID>1250000000</ID>
      <DisplayName>1250000000</DisplayName>
    </Owner>
  </DeleteMarker>
  <Version>
    <Key>example-object-1.jpg</Key>
    <VersionId>MTg0NDUxNzgxMjEzNjYzNzU0MjE</VersionId>
    <IsLatest>false</IsLatest>
    <LastModified>2019-08-16T10:45:43.000Z</LastModified>
    <ETag>&amp;quot;0f0cd12c48979d1bf3f95255a36cb861&amp;quot;</ETag>
    <Size>20</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>1250000000</ID>
      <DisplayName>1250000000</DisplayName>
    </Owner>
  </Version>
  <DeleteMarker>
    <Key>example-object-2.jpg</Key>
    <VersionId>MTg0NDUxNzgxMjEzNTYzMDY3NzY</VersionId>
    <IsLatest>true</IsLatest>
    <LastModified>2019-08-16T10:45:53.000Z</LastModified>
    <Owner>
      <ID>1250000000</ID>
      <DisplayName>1250000000</DisplayName>
    </Owner>
  </DeleteMarker>
  <Version>
    <Key>example-object-2.jpg</Key>
    <VersionId>MTg0NDUxNzgxMjEzNjI5OTc5OTU</VersionId>
```

```
        <IsLatest>false</IsLatest>
        <LastModified>2019-08-16T10:45:46.000Z</LastModified>
        <ETag>&amp;quot;574c289a7906c7d8fecef028216afeca&amp;quot;</ETag>
        <Size>30</Size>
        <StorageClass>STANDARD</StorageClass>
        <Owner>
          <ID>1250000000</ID>
          <DisplayName>1250000000</DisplayName>
        </Owner>
    </Version>
    <Version>
        <Key>example-object-2.jpg</Key>
        <VersionId>MTg0NDUxNzgxMjEzNjgyODc1OTY</VersionId>
        <IsLatest>false</IsLatest>
        <LastModified>2019-08-16T10:45:41.000Z</LastModified>
        <ETag>&amp;quot;51370fc64b79d0d3c7c609635be1c41f&amp;quot;</ETag>
        <Size>20</Size>
        <StorageClass>STANDARD</StorageClass>
        <Owner>
          <ID>1250000000</ID>
          <DisplayName>1250000000</DisplayName>
        </Owner>
    </Version>
</ListVersionsResult>
```

Example 4: The prefix and delimiter Parameters Are Carried (Listing Objects and Subdirectories Under the Root Directory)

Request

```
GET /?versions&prefix=example-folder-1%2F&delimiter=%2F HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Fri, 16 Aug 2019 10:45:53 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKID8A0fBVtYFrNm02oY1g1JQQF0c3JO****&q-sign-tim
e=1565952353;1565959553&q-key-time=1565952353;1565959553&q-header-list=date;host&q-url-pa
ram-list=delimiter;prefix;versions&q-signature=6d7b99a4b379b5fefb8b903ee491bae63590****
Connection: close
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 2682
Connection: close
Date: Fri, 16 Aug 2019 10:45:53 GMT
x-cos-request-id: NWQ1Njg5NjFfMzdiMDJhMDlfODA1NV8yMDI2****
```

```
<ListVersionsResult>
  <Name>examplebucket-1250000000</Name>
  <Prefix>example-folder-1/</Prefix>
  <KeyMarker/>
  <VersionIdMarker/>
  <MaxKeys>1000</MaxKeys>
  <IsTruncated>false</IsTruncated>
  <Delimiter>/</Delimiter>
  <CommonPrefixes>
    <Prefix>example-folder-1/sub-folder-1/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>example-folder-1/sub-folder-2/</Prefix>
  </CommonPrefixes>
  <Version>
    <Key>example-folder-1/example-object-1.jpg</Key>
    <VersionId>MTg0NDUxNzgxMjEzNTY0MzYyNzE</VersionId>
    <IsLatest>true</IsLatest>
    <LastModified>2019-08-16T10:45:53.000Z</LastModified>
    <ETag>&amp;quot;1a54e134fda29e15d225cd226c4b49a3&amp;quot;</ETag>
    <Size>47</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>1250000000</ID>
      <DisplayName>1250000000</DisplayName>
    </Owner>
  </Version>
  <DeleteMarker>
    <Key>example-folder-1/example-object-1.jpg</Key>
    <VersionId>MTg0NDUxNzgxMjEzNjE2MTE1NTI</VersionId>
    <IsLatest>false</IsLatest>
    <LastModified>2019-08-16T10:45:47.000Z</LastModified>
    <Owner>
      <ID>1250000000</ID>
      <DisplayName>1250000000</DisplayName>
    </Owner>
  </DeleteMarker>
  <Version>
    <Key>example-folder-1/example-object-1.jpg</Key>
    <VersionId>MTg0NDUxNzgxMjEzNjcwNjI2MTU</VersionId>
    <IsLatest>false</IsLatest>
    <LastModified>2019-08-16T10:45:42.000Z</LastModified>
    <ETag>&amp;quot;f173c1199e3d3b53dd91223cae16fb42&amp;quot;</ETag>
    <Size>37</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
```

```
            <ID>1250000000</ID>
            <DisplayName>1250000000</DisplayName>
        </Owner>
    </Version>
    <DeleteMarker>
        <Key>example-folder-1/example-object-2.jpg</Key>
        <VersionId>MTg0NDUxNzgxMjEzNTYyOTQ1MDY</VersionId>
        <IsLatest>true</IsLatest>
        <LastModified>2019-08-16T10:45:53.000Z</LastModified>
        <Owner>
            <ID>1250000000</ID>
            <DisplayName>1250000000</DisplayName>
        </Owner>
    </DeleteMarker>
    <Version>
        <Key>example-folder-1/example-object-2.jpg</Key>
        <VersionId>MTg0NDUxNzgxMjEzNjI3NTcyNzU</VersionId>
        <IsLatest>false</IsLatest>
        <LastModified>2019-08-16T10:45:46.000Z</LastModified>
        <ETag>&amp;quot;f43741c189c2142b257767bed5b4ce3a&amp;quot;</ETag>
        <Size>47</Size>
        <StorageClass>STANDARD</StorageClass>
        <Owner>
            <ID>1250000000</ID>
            <DisplayName>1250000000</DisplayName>
        </Owner>
    </Version>
    <Version>
        <Key>example-folder-1/example-object-2.jpg</Key>
        <VersionId>MTg0NDUxNzgxMjEzNjgwNjgwNDk</VersionId>
        <IsLatest>false</IsLatest>
        <LastModified>2019-08-16T10:45:41.000Z</LastModified>
        <ETag>&amp;quot;c9d28698978bb6fef6c1ed1c439a17d3&amp;quot;</ETag>
        <Size>37</Size>
        <StorageClass>STANDARD</StorageClass>
        <Owner>
            <ID>1250000000</ID>
            <DisplayName>1250000000</DisplayName>
        </Owner>
    </Version>
</ListVersionsResult>
```

Example 5: Obtaining the First Page When Pagination Is Required

Request

```
GET /?versions HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Fri, 23 Aug 2019 11:31:31 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKID8A0fBVtYFrNm02oY1g1JQQF0c3JO****&q-sign-tim
e=1566559891;1566567091&q-key-time=1566559891;1566567091&q-header-list=date;host&q-url-pa
ram-list=versions&q-signature=2b146233465c2164c60e0e0b2385f5386a61****
Connection: close
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 372125
Connection: close
Date: Fri, 23 Aug 2019 11:31:32 GMT
x-cos-request-id: NWQ1ZmNlOTRfMTljMDJhMDlfNTg5OV8yZWYz****

<ListVersionsResult>
    <Name>examplebucket-1250000000</Name>
    <Prefix/>
    <KeyMarker/>
    <VersionIdMarker/>
    <MaxKeys>1000</MaxKeys>
    <IsTruncated>true</IsTruncated>
    <NextKeyMarker>example-object-0135.jpg</NextKeyMarker>
    <NextVersionIdMarker>MTg0NDUxNzc1MTM4MjM0ODYyNjQ</NextVersionIdMarker>
    <Version>
        <Key>example-object-0001.jpg</Key>
        <VersionId>MTg0NDUxNzc1MTM4Mjg1NzI5Mjc</VersionId>
        <IsLatest>true</IsLatest>
        <LastModified>2019-08-23T11:31:20.000Z</LastModified>
        <ETag>&amp;quot;3dd9c0ec8b6669abec786e52b64e0497&amp;quot;</ETag>
        <Size>36</Size>
        <StorageClass>STANDARD</StorageClass>
        <Owner>
            <ID>1250000000</ID>
            <DisplayName>1250000000</DisplayName>
        </Owner>
    </Version>
    <DeleteMarker>
        <Key>example-object-0002.jpg</Key>
        <VersionId>MTg0NDUxNzc1MTM4MjY5NDY2MzQ</VersionId>
        <IsLatest>true</IsLatest>
        <LastModified>2019-08-23T11:31:22.000Z</LastModified>
        <Owner>
```

```
        <ID>1250000000</ID>
        <DisplayName>1250000000</DisplayName>
    </Owner>
</DeleteMarker>
<Version>
    <Key>example-object-0002.jpg</Key>
    <VersionId>MTg0NDUxNzc1MTM4MjgzMDg3OTE</VersionId>
    <IsLatest>false</IsLatest>
    <LastModified>2019-08-23T11:31:21.000Z</LastModified>
    <ETag>&amp;quot;626f60ee8f3eb987342554379d63259f&amp;quot;</ETag>
    <Size>36</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
        <ID>1250000000</ID>
        <DisplayName>1250000000</DisplayName>
    </Owner>
</Version>
...
<DeleteMarker>
    <Key>example-object-0004.jpg</Key>
    <VersionId>MTg0NDUxNzc1MTM4MjcwNjg2Mzc</VersionId>
    <IsLatest>false</IsLatest>
    <LastModified>2019-08-23T11:31:22.000Z</LastModified>
    <Owner>
        <ID>1250000000</ID>
        <DisplayName>1250000000</DisplayName>
    </Owner>
</DeleteMarker>
...
<DeleteMarker>
    <Key>example-object-0135.jpg</Key>
    <VersionId>MTg0NDUxNzc1MTM4MjA2NTk1MTc</VersionId>
    <IsLatest>false</IsLatest>
    <LastModified>2019-08-23T11:31:28.000Z</LastModified>
    <Owner>
        <ID>1250000000</ID>
        <DisplayName>1250000000</DisplayName>
    </Owner>
</DeleteMarker>
...
<Version>
    <Key>example-object-0135.jpg</Key>
    <VersionId>MTg0NDUxNzc1MTM4MjM0ODYyNjQ</VersionId>
    <IsLatest>false</IsLatest>
    <LastModified>2019-08-23T11:31:26.000Z</LastModified>
    <ETag>&amp;quot;56d3a714c81ba76baa6a0004126a2718&amp;quot;</ETag>
    <Size>36</Size>
```

```
        <StorageClass>STANDARD</StorageClass>
        <Owner>
            <ID>1250000000</ID>
            <DisplayName>1250000000</DisplayName>
        </Owner>
    </Version>
</ListVersionsResult>
```

Example 6: Obtaining Subsequent Pages When Pagination Is Required (Using the key-marker and version-id-marker Request Parameters)

Request

```
GET /?versions&key-marker=example-object-0135.jpg&version-id-marker=MTg0NDUxNzc1MTM4MjM0ODYyNjQ HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Fri, 23 Aug 2019 11:32:56 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKID8A0fBVtYFrNm02oY1g1JQQF0c3JO****&q-sign-time=1566559976;1566567176&q-key-time=1566559976;1566567176&q-header-list=date;host&q-url-param-list=key-marker;version-id-marker;versions&q-signature=5b0787a354752f3161c75d014b75d9f2bd68****
Connection: close
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 8358
Connection: close
Date: Fri, 23 Aug 2019 11:32:56 GMT
x-cos-request-id: NWQ1ZmNlZThfNjFiMDJhMDlfYTgwOF8yZjIw****

<ListVersionsResult>
    <Name>examplebucket-1250000000</Name>
    <Prefix/>
    <KeyMarker>example-object-0135.jpg</KeyMarker>
    <VersionIdMarker>MTg0NDUxNzc1MTM4MjM0ODYyNjQ</VersionIdMarker>
    <MaxKeys>1000</MaxKeys>
    <IsTruncated>false</IsTruncated>
    <Version>
        <Key>example-object-0135.jpg</Key>
        <VersionId>MTg0NDUxNzc1MTM4MjQ4MDkyNjk</VersionId>
        <IsLatest>false</IsLatest>
        <LastModified>2019-08-23T11:31:24.000Z</LastModified>
        <ETag>&amp;quot;5dd4f1fb98a2a6d74c8482f2856ece6b&amp;quot;</ETag>
        <Size>36</Size>
```

```
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>1250000000</ID>
      <DisplayName>1250000000</DisplayName>
    </Owner>
  </Version>
  <Version>
    <Key>example-object-0135.jpg</Key>
    <VersionId>MTg0NDUxNzc1MTM4MjYwOTQ5MTk</VersionId>
    <IsLatest>false</IsLatest>
    <LastModified>2019-08-23T11:31:23.000Z</LastModified>
    <ETag>&amp;quot;91e59eed612971f0e00ac483bf5c7329&amp;quot;</ETag>
    <Size>36</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>1250000000</ID>
      <DisplayName>1250000000</DisplayName>
    </Owner>
  </Version>
  ...
  <DeleteMarker>
    <Key>example-object-0136.jpg</Key>
    <VersionId>MTg0NDUxNzc1MTM4MTg1MjA4MDA</VersionId>
    <IsLatest>true</IsLatest>
    <LastModified>2019-08-23T11:31:31.000Z</LastModified>
    <Owner>
      <ID>1250000000</ID>
      <DisplayName>1250000000</DisplayName>
    </Owner>
  </DeleteMarker>
  ...
  <DeleteMarker>
    <Key>example-object-0136.jpg</Key>
    <VersionId>MTg0NDUxNzc1MTM4MjA2OTA5MDg</VersionId>
    <IsLatest>false</IsLatest>
    <LastModified>2019-08-23T11:31:28.000Z</LastModified>
    <Owner>
      <ID>1250000000</ID>
      <DisplayName>1250000000</DisplayName>
    </Owner>
  </DeleteMarker>
  ...
  <DeleteMarker>
    <Key>example-object-0137.jpg</Key>
    <VersionId>MTg0NDUxNzc1MTM4MjA2OTQ0MzA</VersionId>
    <IsLatest>false</IsLatest>
    <LastModified>2019-08-23T11:31:28.000Z</LastModified>
```

```
        <Owner>
            <ID>1250000000</ID>
            <DisplayName>1250000000</DisplayName>
        </Owner>
    </DeleteMarker>
    ...
    <Version>
        <Key>example-object-0137.jpg</Key>
        <VersionId>MTg0NDUxNzc1MTM4Mjc1MjQ3OTY</VersionId>
        <IsLatest>false</IsLatest>
        <LastModified>2019-08-23T11:31:22.000Z</LastModified>
        <ETag>&amp;quot;9022b9bf1b1503902d46cbe976c94eea&amp;quot;</ETag>
        <Size>36</Size>
        <StorageClass>STANDARD</StorageClass>
        <Owner>
            <ID>1250000000</ID>
            <DisplayName>1250000000</DisplayName>
        </Owner>
    </Version>
</ListVersionsResult>
```

Example 7: Specifying the Starting Object Using key-marker

Request

```
GET /?versions&key-marker=example-object-0135.jpg HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Fri, 23 Aug 2019 11:32:56 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKID8A0fBVtYFrNm02oY1g1JQQF0c3JO****&q-sign-tim
e=1566559976;1566567176&q-key-time=1566559976;1566567176&q-header-list=date;host&q-url-pa
ram-list=key-marker;versions&q-signature=8b601589aae7ffdb2211694dde4ad73c9634****
Connection: close
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 7111
Connection: close
Date: Fri, 23 Aug 2019 11:32:56 GMT
x-cos-request-id: NWQ1ZmNlZThfNjFiMDJhMDlfYTgwY18yZjRi****

<ListVersionsResult>
    <Name>examplebucket-1250000000</Name>
    <Prefix/>
    <KeyMarker>example-object-0135.jpg</KeyMarker>
```

```
<VersionIdMarker/>
<MaxKeys>1000</MaxKeys>
<IsTruncated>false</IsTruncated>
<DeleteMarker>
  <Key>example-object-0136.jpg</Key>
  <VersionId>MTg0NDUxNzc1MTM4MTg1MjA4MDA</VersionId>
  <IsLatest>true</IsLatest>
  <LastModified>2019-08-23T11:31:31.000Z</LastModified>
  <Owner>
    <ID>1250000000</ID>
    <DisplayName>1250000000</DisplayName>
  </Owner>
</DeleteMarker>
<Version>
  <Key>example-object-0136.jpg</Key>
  <VersionId>MTg0NDUxNzc1MTM4MTg5NjQzNDQ</VersionId>
  <IsLatest>false</IsLatest>
  <LastModified>2019-08-23T11:31:30.000Z</LastModified>
  <ETag>&amp;quot;20f730cd4cab72c0edcbf37c40f9dabe&amp;quot;</ETag>
  <Size>36</Size>
  <StorageClass>STANDARD</StorageClass>
  <Owner>
    <ID>1250000000</ID>
    <DisplayName>1250000000</DisplayName>
  </Owner>
</Version>
<Version>
  <Key>example-object-0136.jpg</Key>
  <VersionId>MTg0NDUxNzc1MTM4MTk3MDAwMDY</VersionId>
  <IsLatest>false</IsLatest>
  <LastModified>2019-08-23T11:31:29.000Z</LastModified>
  <ETag>&amp;quot;339baa9b8d4450e71fc268aaa5fa250e&amp;quot;</ETag>
  <Size>36</Size>
  <StorageClass>STANDARD</StorageClass>
  <Owner>
    <ID>1250000000</ID>
    <DisplayName>1250000000</DisplayName>
  </Owner>
</Version>
<DeleteMarker>
  <Key>example-object-0136.jpg</Key>
  <VersionId>MTg0NDUxNzc1MTM4MjA2OTA5MDg</VersionId>
  <IsLatest>false</IsLatest>
  <LastModified>2019-08-23T11:31:28.000Z</LastModified>
  <Owner>
    <ID>1250000000</ID>
    <DisplayName>1250000000</DisplayName>
```

```
        </Owner>
    </DeleteMarker>
    ...
    <DeleteMarker>
        <Key>example-object-0137.jpg</Key>
        <VersionId>MTg0NDUxNzc1MTM4MjA2OTQ0MzA</VersionId>
        <IsLatest>false</IsLatest>
        <LastModified>2019-08-23T11:31:28.000Z</LastModified>
        <Owner>
            <ID>1250000000</ID>
            <DisplayName>1250000000</DisplayName>
        </Owner>
    </DeleteMarker>
    ...
    <Version>
        <Key>example-object-0137.jpg</Key>
        <VersionId>MTg0NDUxNzc1MTM4Mjc1MjQ3OTY</VersionId>
        <IsLatest>false</IsLatest>
        <LastModified>2019-08-23T11:31:22.000Z</LastModified>
        <ETag>&amp;quot;9022b9bf1b1503902d46cbe976c94eea&amp;quot;</ETag>
        <Size>36</Size>
        <StorageClass>STANDARD</StorageClass>
        <Owner>
            <ID>1250000000</ID>
            <DisplayName>1250000000</DisplayName>
        </Owner>
    </Version>
</ListVersionsResult>
```

Access Control List (ACL)
Setting an ACL for a Bucket

Description

PUT Bucket acl API is used to write an access control list (ACL) for a bucket. You can set the ACL information through "x-cos-acl", "x-cos-grant-read", and "x-cos-grant-full-control" request headers or the request body in XML format.

> Note:
>
> - You can set the ACL information either through request headers or through the request body, otherwise the response returned will conflict.
> - PUT Bucket acl is an overwriting operation. The new ACL will overwrite the old one.
> - Only the Bucket creator has the permission to perform this operation.

Notes

1. You can set the header or the XML body. We recommend you to use only one of them.
2. You can set a folder of private bucket to be public, and then all files in the folder will be public. If you set a folder to be private, the public attributes set in the folder will not take effect.

Request

Request Example

```
PUT /?acl HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Header

Common headers:
This request is implemented by using common request headers. For details on common request headers, see the Common Request Headers section.

Non-common headers:
This request is implemented by using the x-cos-acl header in PUT requests to set the bucket access permissions. Currently, bucket access permissions are divided into three types: public-read-write, public-read, and private. If the

permissions are not set, private is configured by default. The read, write, or read-write permissions can also be granted individually to users. Details are as follows:

Name	Description	Type	Required
x-cos-acl	Defines the ACL attribute of a bucket. Valid values: private, public-read-write, and public-read; default value: private	String	No
x-cos-grant-read	Grants read access to the grantee. Format: x-cos-grant-read: id="[OwnerUin]"	String	No
x-cos-grant-write	Grants write permission to the grantee. Format: x-cos-grant-write: id="[OwnerUin]"	String	No
x-cos-grant-full-control	Grants full permission to the grantee. Format: x-cos-grant-full-control: id="[OwnerUin]"	String	No

Request Body

This request can also be implemented by including specific request parameters in the request body to set the bucket access permissions. You have to choose between including parameters in the request body and including ACL subresources in the request header.
Below is an example with all nodes:

```
<AccessControlPolicy>
  <Owner>
    <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee>
      <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee>
      <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

The data are as follows:

Node Name (Keyword)	Parent Node	Description	Type	Required
AccessControlPolicy	No	Container for storing the result of the GET	Container	Yes

Node Name (Keyword)	Parent Node	Description	Type	Required
		Object ACL		

Content of AccessControlPolicy:

Node Name (Keyword)	Parent Node	Description	Type	Required
Owner	AccessControlPolicy	Information of the object owner	Container	Yes
AccessControlList	AccessControlPolicy	Information about the grantee and permissions	Container	Yes

Content of Owner:

Node Name (Keyword)	Parent Node	Description	Type	Required
ID	AccessControlPolicy.Owner	ID of the Bucket Owner, formatted as qcs::cam::uin/ :uin/. This is the owner account, so and have the same value.	String	Yes

Content of AccessControlList:

Node Name (Keyword)	Parent Node	Description	Type	Required
Grant	AccessControlPolicy.AccessControlList	Permission information for a single bucket. An AccessControlList can have 100 grants.	Container	Yes

Contents of Grant:

Node Name (Keyword)	Parent Node	Description	Type	Required
Grantee	AccessControlPolicy.AccessControlList.Grant	Grantee information. Type is RootAccount.	Container	Yes
Permission	AccessControlPolicy.AccessControlList.Grant	Specifies the permissions granted to the grantee, with enum values: READ,	String	Yes

Node Name (Keyword)	Parent Node	Description	Type	Required
		FULL_CONTROL.		

Contents of Grantee:

Node Name (Keyword)	Parent Node	Description	Type	Required
ID	AccessControlPolicy.AccessControlList.Grant.Grantee	ID of the User, Format: qcs::cam::uin/<OwnerUin>:uin/<SubUin> . The ID should be that of the root account. Therefore, and have the same value. You can also use anyone (representing all types of users) to replace uin/ and uin/.	String	Yes

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Common Response Headers.

Special Response Headers

There are no special response headers for this response operation.

Response Body

The response body is empty.

Error Analysis

The following describes some special but common errors that may occur with this request:

Error Code	Description	HTTP Status Code

Error Code	Description	HTTP Status Code
InvalidDigest	400 Bad Request	Content-MD5 provided by the user is inconsistent with Content-MD5 in the body calculated by COS.
MalformedXML	400 Bad Request	The XML syntax of the request body is incorrect. Please compare it carefully with the restful API document.
InvalidArgument	400 Bad Request	Parameter error, see the error message for details.

Practical Case

Request

```
PUT /?acl HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Fri, 25 Feb 2017 04:10:22 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=1484724784;32557620784&q-key-time=1484724784;32557620784&q-header-list=host&q-url-param-list=acl&q-signature=785d9075b8154119e6a075713c1b9e56ff0bddfc
Content-Length: 229
Content-Type: application/x-www-form-urlencoded

<AccessControlPolicy>
  <Owner>
    <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee>
      <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee>
      <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

Response

```
HTTP/1.1 200 OK
```

Content-Type: application/xml
Content-Length: 0
Connection: keep-alive
Date: Fri, 25 Feb 2017 04:10:22 GMT
x-cos-request-id: NTg3ZjFjMmJfOWIxZjRlXzZmNDhfMjIw

Querying an ACL of a Bucket

Description

The GET Bucket acl API is used to query the access control list (ACL) of a bucket.

Request

Request Example

```
GET /?acl HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Header

Common Headers

This request uses common request headers. For details, see Public Request Headers.

Non-common Headers

This request has no special request header information.

Request Body

The request does not have a request body.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Common Response Headers.

Special Response Headers

There are no special response headers for this response operation.

Response Body

The response body is returned as application/xml. An example including the complete node data is shown below:

```
<AccessControlPolicy>
  <Owner>
    <ID>qcs::cam::uin/1250000000:uin/1250000000</ID>
    <DisplayName>qcs::cam::uin/1250000000:uin/1250000000</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:type=\"CanonicalUser
\">
        <ID>qcs::cam::uin/1250000000:uin/1250000000</ID>
        <DisplayName>qcs::cam::uin/1250000000:uin/1250000000</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

The data are as follows:

Node Name (Keyword)	Parent Node	Description	Type
AccessControlPolicy	No	Container for storing the result of the GET Bucket acl	Container

Content of AccessControlPolicy:

Node Name (Keyword)	Parent Node	Description	Type
Owner	AccessControlPolicy	Information of the object owner	Container
AccessControlList	AccessControlPolicy	Information about the grantee and permissions	Container

Content of Owner:

Node Name (Keyword)	Parent Node	Description	Type
ID	AccessControlPolicy.Owner	ID of the Bucket Owner,	String

Node Name (Keyword)	Parent Node	Description	Type
		formatted as qcs::cam::uin/ :uin/. This is the root account, so and have the same value.	
DisplayName	AccessControlPolicy.Owner	Name of the Bucket Owner	String

Content of AccessControlList:

Node Name (Keyword)	Parent Node	Description	Type
Grant	AccessControlPolicy.AccessControlList	Permission information for a single bucket. An AccessControlList can have 100 grants.	Container

Contents of Grant:

Node Name (Keyword)	Parent Node	Description	Type
Grantee	AccessControlPolicy.AccessControlList.Grant	Grantee Information. If type is RootAccount or CanonicalUser, the account in the ID is the root account.	Container
Permission	AccessControlPolicy.AccessControlList.Grant	Specifies the permissions granted to the grantee, with enum values: READ, FULL_CONTROL.	String

Contents of Grantee:

Node Name (Keyword)	Parent Node	Description	Type
ID	AccessControlPolicy.Owner	ID of the user, which is the root account, formatted as qcs::cam::uin/:uin/ or qcs::cam::anyone:anyone (referring to all users).	String

Node Name (Keyword)	Parent Node	Description	Type
DisplayName	AccessControlPolicy.Owner	User name	String

Error Code

This request returns common error responses and error codes. For more information, see Error Codes.

Practical Case

Request

```
GET /?acl HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Fri, 10 Mar 2016 09:45:46 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84213027;32557109027&q-key-time=1484213027;32557109027&q-header-list=host&q-url-param-list
=acl&q-signature=dcc1eb2022b79cb2a780bf062d3a40e120b40652
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 266
Connection: keep-alive
Date: Fri, 10 Mar 2016 09:45:46 GMT
x-cos-request-id: NTg3NzRiMjVfYmRjMzVfMTViMl82ZGZmNw==
<AccessControlPolicy>
  <Owner>
    <ID>qcs::cam::uin/1250000000:uin/1250000000</ID>
    <DisplayName>qcs::cam::uin/1250000000:uin/1250000000</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:type=\"CanonicalUser
\">
        <ID>qcs::cam::uin/1250000000:uin/1250000000</ID>
        <DisplayName>qcs::cam::uin/1250000000:uin/1250000000</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

Cross-Origin Resource Sharing (CORS) Setting CORS Configuration

Description

The PUT Bucket API is used to configure CORS for a bucket. You need to pass in an XML configuration file within 64 KB. By default, the owner of the bucket has the permission to access the API, and can grant permissions to other users.

Request

Request sample:

```
PUT /?cors HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Content-Length: length
Content-Type: application/xml
Content-MD5: MD5
Authorization: Auth String

<XML file>
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Line

```
PUT /?cors HTTP/1.1
```

This API accepts `PUT` requests.

Request Header

Common Headers

This request uses common request headers. For details, see Common Request Headers.

Non-common Headers

This request has no special request header information.

Request Body

The request body is a CORS rule.

```
<?xml version="1.0" encoding="UTF-8" ?>
<CORSConfiguration>
  <CORSRule>
    <ID>string</ID>
    <AllowedOrigin>string</AllowedOrigin>
    <AllowedMethod>string</AllowedMethod>
    <AllowedHeader>string</AllowedHeader>
    <MaxAgeSeconds>0</MaxAgeSeconds>
    <ExposeHeader>string</ExposeHeader>
  </CORSRule>
</CORSConfiguration>
```

The data are described as follows:

Node Name (Keyword)	Parent Node	Description	Type	Required
CORSConfiguration	No	All information about CORS rule. You can configure up to 100 CORSRule.	Container	Yes

Content of CORSConfiguration:

Node Name (Keyword)	Parent Node	Description	Type	Required
CORSRule	CORSConfiguration	All information about CORS rule. You can configure up to 100 CORSRule.	Container	Yes

Content of CORSRule:

Node Name (Keyword)	Parent Node	Description	Type	Required
ID	CORSConfiguration.CORSRule	ID of the CORSRule, not required.	string	Yes
AllowedOrigin	CORSConfiguration.CORSRule	Allowed access sources. The wildcard * is supported. Format: protocol://domain name[:port], such as,	strings	Yes

Node Name (Keyword)	Parent Node	Description	Type	Required
		http://www.test.com.		
AllowedMethod	CORSConfiguration.CORSRule	Allowed HTTP operations. Enumerated values: GET, PUT, HEAD, POST, and DELETE	strings	Yes
AllowedHeader	CORSConfiguration.CORSRule	When an OPTIONS request is sent, the browser will let the server know what custom HTTP headers will be used in the actual request. You can set it to * to allow all domains.	strings	Yes
MaxAgeSeconds	CORSConfiguration.CORSRule	Validity period of the result obtained by the OPTIONS request.	integer	Yes
ExposeHeader	CORSConfiguration.CORSRule	User-defined header information from the server side that the browser can receive.	strings	Yes

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Public Response Header.

Special Response Headers

This request has no special response header information.

Response Body

The response body is empty.

Error Code

Error Code	Description	HTTP Status Code
SignatureDoesNotMatch	If this error code is returned, it means the provided signature is incorrect.	403 Forbidden
NoSuchBucket	If this error code is returned, it means the bucket you want to add the rule to is not found.	404 Not Found

Practical Case

Request

```
PUT /?cors HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Fri, 10 Mar 2017 09:45:46 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84814927;32557710927&q-key-time=1484814927;32557710927&q-header-list=host&q-url-param-list
=cors&q-signature=8b9f05dabce2578f3a79d732386e7cbade9033e3
Content-Type: application/xml
Content-Length: 280

<CORSConfiguration>
   <CORSRule>
      <ID>1234</ID>
      <AllowedOrigin>http: //www.test.com</AllowedOrigin>
      <AllowedMethod>PUT</AllowedMethod>
      <AllowedHeader>x-cos-meta-test</AllowedHeader>
      <MaxAgeSeconds>500</MaxAgeSeconds>
      <ExposeHeader>x-cos-meta-test1</ExposeHeader>
   </CORSRule>
</CORSConfiguration>
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Connection: keep-alive
Date: Fri, 10 Mar 2017 09:45:46 GMT
x-cos-request-id: NTg4MDdiZWRfOWExZjRlXzQ2OWVfZGY0
```

Querying CORS Configuration

Description

The GET Bucket cors API allows the bucket owner to s et the CORS for a bucket. (cors is a W3C standard, the full name is Cross-origin resource sharing). By default, the owner of the bucket has the permission to access the API, and can grant permissions to other users.

Request

Request Example

```
GET /?cors HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Header

Common Headers

This request uses common request headers. For details, see Common Request Headers.

Non-common Headers

This request has no special request header information.

Request Body

The request body of this request is empty.

Response

Response Headers

Common Response Headers

This response contains common response headers. For details, see Common Response Headers.

Special Response Headers

There are no special response headers for this response operation.

Response Body

The CORS has been configured successfully.

```
<?xml version="1.0" encoding="UTF-8" ?>
<CORSConfiguration>
  <CORSRule>
    <ID>bucketid</ID>
    <AllowedOrigin>http://www.test.com</AllowedOrigin>
    <AllowedMethod>PUT</AllowedMethod>
    <AllowedHeader>x-cos-meta-test</AllowedHeader>
    <ExposeHeader>x-cos-meta-test1</ExposeHeader>
    <MaxAgeSeconds>500</MaxAgeSeconds>
  </CORSRule>
</CORSConfiguration>
```

The data are described as follows:

Node Name (Keyword)	Parent Node	Description	Type	Required
CORSConfiguration	No	All information about CORS rule. You can configure up to 100 CORSRule.	Container	Yes

Content of CORSConfiguration:

Node Name (Keyword)	Parent Node	Description	Type	Required
CORSRule	CORSConfiguration	All information about CORS rule. You can configure up to 100 CORSRule.	Container	Yes

Content of CORSRule:

Node Name (Keyword)	Parent Node	Description	Type	Required
ID	CORSConfiguration.CORSRule	ID of the CORSRule, not required.	string	Yes
AllowedOrigin	CORSConfiguration.CORSRule	Allowed access source. The wildcard * is	strings	Yes

Node Name (Keyword)	Parent Node	Description	Type	Required
		supported. Format: protocol://domain name[:port], such as http://www.test.com .		
AllowedMethod	CORSConfiguration.CORSRule	Allowed HTTP operations. Enumerated values: GET, PUT, HEAD, POST, and DELETE	strings	Yes
AllowedHeader	CORSConfiguration.CORSRule	When an OPTIONS request is sent, the browser will let the server know what custom HTTP headers will be used in the actual request. You can set it to * to allow all domains.	strings	Yes
MaxAgeSeconds	CORSConfiguration.CORSRule	Validity period of the result obtained by the OPTIONS request.	integer	Yes
ExposeHeader	CORSConfiguration.CORSRule	User-defined header information from the server side that the browser can receive.	strings	Yes

Error Code

This request returns common error responses and error codes. For more information, see Error Codes.

Practical Case

Request

```
GET /?cors HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Wed, 28 Oct 2016 21:32:00 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84815944;32557711944&q-key-time=1484815944;32557711944&q-header-list=host&q-url-param-list
=cors&q-signature=a2d28e1b9023d09f9277982775a4b3b705d0e23e
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 345
Connection: keep-alive
Date: Wed, 28 Oct 2016 21:32:00 GMT
x-cos-request-id: NTg4MDdlNGZfNDYyMDRlXzM0YWFfZTBh

<CORSConfiguration>
  <CORSRule>
    <ID>bucketid</ID>
    <AllowedOrigin>http://www.test.com</AllowedOrigin>
    <AllowedMethod>PUT</AllowedMethod>
    <AllowedHeader>x-cos-meta-test</AllowedHeader>
    <ExposeHeader>x-cos-meta-test1</ExposeHeader>
    <MaxAgeSeconds>500</MaxAgeSeconds>
  </CORSRule>
</CORSConfiguration>
```

Deleting CORS Configuration

Description

The DELETE Bucket cors API is used to delete the CORS configuration.

Request

Syntax example:

```
DELETE /?cors HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Line

```
DELETE /?cors HTTP/1.1
```

This API accepts DELETE requests.

Request Header

Common Headers

This request uses common request headers. For details, see Common Request Headers.

Non-common Headers

This request has no special request header information.

Request Body

The request does not have a request body.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Public Response Header.

Special Response Headers

There are no special response headers for this response operation.

Response Body

The response body is empty.

Practical Case

Request

```
DELETE /?cors HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Tue, 23 Oct 2016 21:32:00 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84816036;32557712036&q-key-time=1484816036;32557712036&q-header-list=host&q-url-param-list
=cors&q-signature=e92eecbf0022fe7e5fd39b2c500b22da062be50a
```

Response

```
HTTP/1.1 204 No Content
Content-Type: application/xml
Content-Length: 405
Connection: keep-alive
Date: Tue, 23 Oct 2016 21:32:00 GMT
x-cos-request-id: NTg4MDdlYWNfOTgxZjRlXzZhYTlfZjAz
x-cos-trace-id: OGVmYzZiMmQzYjA2OWNhODk0NTRkMTBiOWVmMDAxODczNTBmNjMwZmQ0MTZk
Mjg0NjlkNTYyNmY4ZTRkZTk0N2M2MTdkZGZlMGNhOWQyYjk3MWNmNWNkYzFhMjQzNzRiZTE1Njg
zNzFhOGI5M2EwZDMyNGM4Y2ZmMzhiNTllMjk=
```

Lifecycle
Setting the Bucket Lifecycle

Description

COS allows you to manage the lifecycle of objects in buckets through lifecycle configuration, which contains one or more rulesets that will be applied to a set of object rules (where each rule defines an operation in COS).
These operations include the following two types:

- Conversion: Defines the time when an object is converted to another storage type. For example, you can convert an object to the infrequent access storage (STANDARD_IA, suitable for infrequent access) 30 days after creation. The data can also be stored in the archive storage (ARCHIVE, lower costs and available in China). For specific parameters, see the Transition item in the sample request description.
- Expiration: Specifies the expiration time of an object. COS automatically deletes expired objects.

Notes

PUT Bucket lifecycle is used to create a new lifecycle configuration for a Bucket. If a lifecycle configuration has already been set for the bucket, the new configuration created with this API will overwrite the existing one.

Request

Request Example

```
PUT /?lifecycle HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Content-Length: length
Date: GMT Date
Authorization: Auth String
Content-MD5: MD5
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Header

Common Headers

This request uses common request headers. For details, see Public Request Headers.

Non-common Headers

Required headers:
This request is implemented by using the following required headers.

Name	Description	Type	Required
Content-MD5	Base64-encoded 128-bit content MD5 checksum as defined in RFC 1864. This header is used to verify whether the file content has changed.	String	Yes

Request Body

The specific node content of the API request body is:

```
<LifecycleConfiguration>
  <Rule>
    <ID></ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
```

```
    <Status></Status>
    <Transition>
      <Days></Days>
      <StorageClass></StorageClass>
    </Transition>
    <NoncurrentVersionExpiration>
      <NoncurrentDays></NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
  <Rule>
    <ID></ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Status></Status>
    <Transition>
      <Days></Days>
      <StorageClass></StorageClass>
    </Transition>
    <NoncurrentVersionTransition>
      <NoncurrentDays></NoncurrentDays>
      <StorageClass></StorageClass>
    </NoncurrentVersionTransition>
  </Rule>
  <Rule>
    <ID></ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Status></Status>
    <Expiration>
      <ExpiredObjectDeleteMarker></ExpiredObjectDeleteMarker>
    </Expiration>
    <NoncurrentVersionExpiration>
      <NoncurrentDays></NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
</LifecycleConfiguration>
```

The specific contents are described as follows:

Node Name (Keyword)	Parent Node	Description	Type	Required
LifecycleConfiguration	No	Lifecycle configuration	Container	Yes
Rule	LifecycleConfiguration	Rule description	Container	Yes
Filter	LifecycleConfiguration.Rule	Identifies objects that a lifecycle rule applies to.	Container	Yes
Status	LifecycleConfiguration.Rule	Indicates whether the rule is enabled. Enumerated values: Enabled, Disabled	Container	Yes
ID	LifecycleConfiguration.Rule	A unique identifier for the rule. It can be up to 255 characters.	String	No

Node Name (Keyword)	Parent Node	Description	Type	Required
And	LifecycleConfiguration.Rule.Filter	Used to combine Prefix	Container	No
Prefix	LifecycleConfiguration.Rule.Filter or LifecycleConfiguration.Rule.Filter.And	Matching prefix for the rule. It specifies objects that the lifecycle rule applies to. There can be one Prefix at most.	Container	No
Expiration	LifecycleConfiguration.Rule	Expiration attributes of the rule	Container	No
Transition	LifecycleConfiguration.Rule	Transition attributes of the rule. Specifies when to transition the object to Standard_IA or Archive	Container	No
Days	LifecycleConfiguration.Rule.Transition or Expiration	Specifies the number of days between the date an object was last modified and the date when the operation corresponding to the rule is performed. If it is a Transition operation, this value should be a non-negative integer. If it is an Expiration operation, this value should be a positive integer. The maximum value is 3650 (days).	Integer	No
Date	LifecycleConfiguration.Rule.Transition or Expiration	Specifies when the operation corresponding to the rule is performed.	String	No
ExpiredObjectDeleteMarker	LifecycleConfiguration.Rule.Expiration	Indicates whether the delete marker of an expired object will be removed. Enumerated values: true,	String	No

Node Name (Keyword)	Parent Node	Description	Type	Required
		false		
AbortIncompleteMultipartUpload	LifecycleConfiguration.Rule	Specifies the maximum time before a multipart upload is aborted.	Container	No
DaysAfterInitiation	LifecycleConfiguration.Rule.AbortIncompleteMultipartUpload	Specifies the number of days within which the multipart upload must be completed after it starts.	Integer	Yes
NoncurrentVersionExpiration	LifecycleConfiguration.Rule	Specifies when noncurrent object versions shall expire.	Container	No
NoncurrentVersionTransition	LifecycleConfiguration.Rule	Specifies when to transition objects of noncurrent versions to STANDARD_IA or ARCHIVE.	Container	No
NoncurrentDays	LifecycleConfiguration.Rule.NoncurrentVersionExpiration or NoncurrentVersionTransition	Specifies the number of days between the date when an object becomes noncurrent and the date when the operation corresponding to a rule is performed. If it is a Transition operation, this value should be a non-negative integer. If it is an Expiration operation, this value should be a positive integer. The maximum value is 3650 (days).	Integer	No
StorageClass	LifecycleConfiguration.Rule.Transition or NoncurrentVersionTransition	Specifies the storage class of the transitioned object. Enumerated values: STANDARD_IA, ARCHIVE	String	Yes

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Common Response Headers.

Special Response Headers

There are no special response headers for this response operation.

Response Body

The response body is empty.

Error Code

The following table describes some special and common errors that may occur with this request. For detailed error causes, check the returned message. For details on error codes related to COS or the complete list of errors for the product, see Error Codes.

Error Code	HTTP Status Code	Description
NoSuchBucket	404 Not Found	The bucket does not exist.
MalformedXML	400 Bad Request	The XML syntax of the request body is incorrect. Please compare it carefully with the restful API document.
InvalidRequest	400 Bad Request	The request is illegal. If Conflict lifecycle rule is displayed in the error description, it means that multiple rules in the XML data have conflicting parts.
InvalidArgument	400 Bad Request	The request parameter is illegal. If the error description shows Rule ID must be unique. Found same ID for more than one rule, it means that multiple rules have the same ID field.

Practical Case

Request

```
PUT /?lifecycle HTTP/1.1
Host:<BucketName-APPID>.<Endpoint>
Date: Wed, 16 Aug 2017 11:59:33 GMT
Authorization:q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=1502855771;1502935771&q-key-time=1502855771;1502935771&q-header-list=content-md5;host&q-url-param-list=lifecycle&q-signature=f3aa2c708cfd8d4d36d658de56973c9cf1c24654
Content-MD5: LcNUuow8OSZMrEDnvndw1Q==
Content-Length: 348
Content-Type: application/x-www-form-urlencoded

<LifecycleConfiguration>
  <Rule>
    <ID>id1</ID>
    <Filter>
      <Prefix>documents/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>100</Days>
      <StorageClass>ARCHIVE</StorageClass>
    </Transition>
  </Rule>
  <Rule>
    <ID>id2</ID>
    <Filter>
      <Prefix>logs/</Prefix>
```

```
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <Days>10</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Date: Wed, 16 Aug 2017 11:59:33 GMT
x-cos-request-id: NTk5NDMzYTRfMjQ4OGY3Xzc3NGRfMWY=
```

Querying the Bucket Lifecycle Configuration

Description

The GET Bucket lifecycle API is used to query the lifecycle configuration of a bucket. If no lifecycle rule is configured for the bucket, NoSuchLifecycleConfiguration will be returned.

Request

Request Example

```
GET /?lifecycle HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Header

Common Headers

This request uses common request headers. For details, see Public Request Headers.

Non-common Headers

This request has no special request header information.

Request Body

The request does not have a request body.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Common Response Headers.

Special Response Headers

There are no special response headers for this response operation.

Response Body

The content and meaning of each element in the response body are the same as those in the request body of the PUT Bucket lifecycle API. For details, see the description of the request body in Setting Lifecycle.

Error Code

This request returns common error responses and error codes. For more information, see Error Codes.

Practical Case

Request

```
GET /?lifecycle HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Wed, 16 Aug 2017 12:23:54 GMT
Authorization:q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=15
02857357;1502937357&q-key-time=1502857357;1502937357&q-header-list=host&q-url-param-list=li
fecycle&q-signature=da155cda3461bee7422ee95367ac8013ef847e02
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 312
Date: Wed, 16 Aug 2017 12:23:54 GMT
x-cos-request-id: NTk5NDM5NWFfMjQ4OGY3Xzc3NGRfMjA=

<LifecycleConfiguration>
  <Rule>
    <ID>id1</ID>
    <Filter>
      <Prefix>documents/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>100</Days>
      <StorageClass>STANDARD_IA</StorageClass>
    </Transition>
  </Rule>
  <Rule>
    <ID>id2</ID>
```

```
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <Days>10</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

Deleting the Bucket Lifecycle Configuration

Description

The DELETE Bucket lifecycle API is used to delete the lifecycle configuration set for a bucket. If no lifecycle rule is configured for the bucket, NoSuchLifecycleConfiguration will be returned.

Request

Request Example

```
DELETE /?lifecycle HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Header

Common Headers

This request uses common request headers. For details, see Public Request Headers.

Non-common Headers

This request has no special request header information.

Request Body

The request does not have a request body.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Common Response Headers.

Special Response Headers

There are no special response headers for this response operation.

Response Body

The response body is empty.

Error Code

This request returns the following error codes. For more information, see Error Codes.

Error Code	Description	HTTP Status Code
None	The deletion is successful and the response body returns empty.	204 No Content
NoSuchBucket	If this error code is returned, it means there is no such bucket to access.	404 Not Found

Practical Case

Request

```
DELETE /?lifecycle HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Wed, 16 Aug 2017 12:59:09 GMT
Authorization:q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=15
02859472;1502939472&q-key-time=1502859472;1502939472&q-header-list=host&q-url-param-list=li
fecycle&q-signature=49c1414c700643f11139219384332a3ec4e9485b
```

Response

```
HTTP /1.1 204 No Content
Content-Type: application/xml
Date: Wed, 16 Aug 2017 12:59:09 GMT
x-cos-request-id: NTk5NDQxOWNfMjQ4OGY3Xzc3NGRfMjE=
```

Bucket Policy
Setting a Bucket Policy

Description

The PUT Bucket policy API is used to write a permission policy for a bucket. The policy passed in this API will overwrite the existing one (if any) in the bucket.

Request

Request Example

```
PUT /?policy HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: date
Content-Type:application/json
Content-MD5:MD5
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Header

Common Headers

This request uses common request headers. For details, see Public Request Headers.

Request Parameters

No special request parameters are included.

Request Body

For more information about the elements used in access policy settings, see Access Policy Language Overview.

```
{
   "Statement": [
      {
         "Principal": {
```

```
            "qcs": [
                "qcs::cam::uin/${owner_uin}:uin/${sub_uin}"
            ]
        },
        "Effect": "${effect}",
        "Action": [
            "name/cos:${action}"
        ],
        "Resource": [
            "qcs::cos:${region}:uid/${appid}:${bucket}-${appid}/*"
        ]
    }
  ],
  "version": "2.0"
}
```

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Common Response Headers.

Special Response Headers

This request has no special response header information.

Response Body

The response body is empty.

Error Code

No special error code is returned. For common error codes, see Error Codes.

Practical Case

Request

```
PUT /?policy HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Authorization:q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84813288;32557709288&q-key-time=1484813288;32557709288&q-header-list=host&q-url-param-list
=policy&q-signature=05f7fc936369f910a94a0c815e1f1752f034d47a
```

```
Content-Type: application/json
Content-Length: 233

{
  "Statement": [
    {
      "Principal": {
        "qcs": [
          "qcs::cam::uin/1250000000:uin/1250000000"
        ]
      },
      "Effect": "allow",
      "Action": [
        "name/cos:GetBucket"
      ],
      "Resource": [
        "qcs::cos:ap-city:uid/1250000000:examplebucket-1250000000/*"
      ]
    }
  ],
  "version": "2.0"
}
```

Response

```
HTTP/1.1 204 No Content
Content-Type: application/xml
Content-Length: 0
Connection: keep-alive
Date: Thu Jan 19 16:19:22 2017
x-cos-request-id: NTg4MDc2OGFfNDUyMDRlXzc3NTlfZTc4
```

Querying a Bucket Policy

Description

The GET Bucket policy API is used to read the permission policy of a bucket.

Request

Request Example

```
GET /?policy HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date:date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Header

Common Headers

This request uses common request headers. For details, see Public Request Headers.

Non-common Headers

This request has no special request header information.

Request Body

The request body of this request is empty.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Common Response Headers.

Special Response Headers

There are no special response headers for this response operation.

Response Body

The permission policy is returned in the response body.

```
{
  "Statement": [
    {
      "Principal": {
        "qcs": [
          "qcs::cam::uin/100000000001:uin/100000000001"
        ]
      },
      "Effect": "allow",
      "Action": [
        "name/cos:GetBucket"
      ],
      "Resource": [
        "qcs::cos:ap-city:uid/1250000000:examplebucket-1250000000/*"
      ]
    }
  ],
  "version": "2.0"
}
```

Error Code

This request returns common error responses and error codes. For more information, see Error Codes.

Practical Case

Request

```
GET /?policy HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Authorization:q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84814099;32557710099&q-key-time=1484814099;32557710099&q-header-list=host&q-url-param-list
=policy&q-signature=0523d7c6305b6676611c44798d2c48b659e68869
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 237
Connection: keep-alive
```

Date: Thu Jan 19 16:21:46 2017
x-cos-request-id: NTg4MDc3MWFfOWIxZjRlXzZmNDVfZTBl


```
{
  "Statement": [
    {
      "Principal": {
        "qcs": [
          "qcs::cam::uin/100000000001:uin/100000000001"
        ]
      },
      "Effect": "allow",
      "Action": [
        "name/cos:GetBucket"
      ],
      "Resource": [
        "qcs::cos:ap-city:uid/1250000000:examplebucket-1250000000/*"
      ]
    }
  ],
  "version": "2.0"
}
```

Deleting a Bucket Policy

Description

The DELETE Bucket policy request can be used to delete the permission policy for a bucket.

> Note:
>
> Only the bucket owner has permission to initiate this request. If the permission policy does not exist, a 204 No Content response is returned.

Request

Request Example

```
DELETE /?policy HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Header

Common Headers

This request operation is implemented by using the public request header. For details about the public request header, see Common Request Headers documentation.

Non-common Headers

This request has no special request header information.

Request Body

The request does not have a request body.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Common Response Headers.

Special Response Headers

There are no special response headers for this response operation.

Response Body

The response body is empty.

Practical Case

Request

```
DELETE /?policy HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Authorization:q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84814613;32557710613&q-key-time=1484814613;32557710613&q-header-list=host&q-url-param-list
=policy&q-signature=57c9a3f67b786ddabd2c208641944ec7f9b02f98
```

Response

```
HTTP/1.1 204 No Content
Content-Type: application/xml
Content-Length: 0
Connection: keep-alive
Date: Thu Jan 19 16:30:21 2017
x-cos-request-id: NTg4MDc5MWRfNDQyMDRlXzNiMDRfZTEw
```

Hotlink Protection (Referer)
Setting a Bucket Referer

Description

The PUT Bucket referer API is used to set a Referer allowlist or blocklist for a bucket.

Request

Request Example

```
PUT /?referer HTTP 1.1
Host: <BucketName-APPID>.<Endpoint>
Date:GMT Date
Authorization: Auth String
Content-Length:length
Content-MD5:MD5
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Header

Common Headers

This request uses common request headers. For details, see Public Request Headers.

Non-common Headers

Required Headers

Name	Description	Type	Required
Content-Length	Length (bytes) of HTTP request content as defined in RFC 2616.	String	Yes
Content-MD5	MD5 checksum value of the 128-bit content encoded with Base64 defined in RFC 1864. This header is used to check whether the file content is changed.	String	Yes

Request Body

The specific node content of the request body of this request is as follows.

```
<RefererConfiguration>
  <Status>Enabled</Status>
  <RefererType>White-List</RefererType>
  <DomainList>
    <Domain>*.domain1.com</Domain>
    <Domain>*.domain2.com</Domain>
  </DomainList>
  <EmptyReferConfiguration>Allow</EmptyReferConfiguration>
</RefererConfiguration>
```

The specific contents are described as follows:

Name	Parent Node	Description	Type	Required
RefererConfiguration	No	Anti-hotlinking configuration information	Container	Yes
Status	RefererConfiguration	Whether to enable anti-hotlinking, enumeration value: Enabled, Disabled.	String	Yes
RefererType	RefererConfiguration	Anti-hotlinking type, enumeration value: Black-List, White-List	String	Yes
DomainList	RefererConfiguration	Valid domain name list. Supports multiple domain names (prefix matching). Supports domain names and IP addresses with ports. Supports the wildcard * as a wildcard for second-level domain names or multi-level domain names.	Container	Yes

Name	Parent Node	Description	Type	Required
Domain	DomainList	Single valid domain name For example, www.test.com/example , 192.168.1.2:8080 , and *.test.com .	String	Yes
EmptyReferConfiguration	RefererConfiguration	Whether to allow access with a null Referer. Enumeration value: Allow, Deny. The default value is Deny.	String	No

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

The response body is empty.

Error Code

This request has no error. For all errors, see Error Codes documentation.

Practical Case

Request

```
PUT /?referer HTTP 1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Fri, 25 Feb 2017 04:10:22 GMT
Authorization:q-sign-algorithm=sha1&q-ak=AKIDZfbOAo7cIlgPvF9cXFrJD0a1ICvR****&q-sign-time=1547105134;32526689134&q-key-time=1547105134;32620001134&q-header-list=content-md5;content-type;host&q-url-param-list=referer&q-signature=0f7fef5b1d2180deaf6f92fa2ee0cf87ae83f****
Content-MD5: kOz7g54LMJZjxKs070V9jQ==
Content-Type: application/xml
```

```
<RefererConfiguration>
  <Status>Enabled</Status>
  <RefererType>White-List</RefererType>
  <DomainList>
    <Domain>*.domain1.com</Domain>
    <Domain>*.domain2.com</Domain>
  </DomainList>
  <EmptyReferConfiguration>Allow</EmptyReferConfiguration>
</RefererConfiguration>
```

Response

```
HTTP/1.1 200 OK
Content-Length: 0
Connection: keep-alive
Date: Fri, 25 Feb 2017 04:10:22 GMT
x-cos-request-id: NTg3ZjFjMmJfOWIxZjRlXzZmNDhf****
```

Querying a Bucket Referer

Description

The GET Bucket referer API is used to read the bucket referer allowlist or blocklist.

Request

Request Example

```
GET /?referer HTTP 1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (For details, see Request Signature documentation).

Request Header

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

The request body of this request is empty.

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

The response body is returned as application/xml data, and the content containing the complete node data is shown below.

```
<RefererConfiguration>
    <Status>Enabled</Status>
    <RefererType>White-List</RefererType>
```

```
    <DomainList>
      <Domain>*.domain1.com</Domain>
      <Domain>*.domain2.com</Domain>
    </DomainList>
    <EmptyReferConfiguration>Allow</EmptyReferConfiguration>
  </RefererConfiguration>
```

The data are as follows:

Name	Parent Node	Description	Type	Required
RefererConfiguration	No	Anti-hotlinking configuration information	Container	Yes
Status	RefererConfiguration	Whether to enable hotlink protection. Enumerated values: Enabled and Disabled	String	Yes
RefererType	RefererConfiguration	Anti-hotlink type, enumeration value: Black-List, White-List	String	Yes
DomainList	RefererConfiguration	Valid domain name list. Supports multiple domain names (prefix matching). Supports domain names and IP addresses with ports. Supports the wildcard * for second-level domain names or multi-level domain names.	Container	Yes
Domain	DomainList	A single valid domain, such as www.test.com/ example , 192.168.1.2:8080 , and *.test.com .	String	Yes
EmptyReferConfiguration	RefererConfiguration	Whether to allow access with a null Referer. Enumeration value: Allow, Deny. The default value is	String	No

Name	Parent Node	Description	Type	Required
		Deny.		

Error Code

This request has no error. For all errors, see Error Codes documentation.

Practical Case

Request

```
GET /?referer HTTP 1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Fri, 25 Feb 2017 04:10:22 GMT
Authorization:q-sign-algorithm=sha1&q-ak=AKIDZfbOAo7cIlgPvF9cXFrJD0a1ICvR****&q-sign-time=1547105134;32526689134&q-key-time=1547105134;32620001134&q-header-list=content-md5;content-type;host&q-url-param-list=referer&q-signature=0f7fef5b1d2180deaf6f92fa2ee0cf87ae83****
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 260
Connection: keep-alive
Date: Fri, 25 Feb 2017 04:10:22 GMT
x-cos-request-id: NTg3ZjFjMmJfOWIxZjRlXzZmNDhf****

<RefererConfiguration>
    <Status>Enabled</Status>
    <RefererType>White-List</RefererType>
    <DomainList>
        <Domain>*.domain1.com</Domain>
        <Domain>*.domain2.com</Domain>
    </DomainList>
    <EmptyReferConfiguration>Allow</EmptyReferConfiguration>
</RefererConfiguration>
```

Tag
Setting Bucket Tags

Description

COS allows you to set tags for an existing bucket. The PUT Bucket tagging API is used to set key-value pairs as bucket tags, helping you manage existing bucket resources and conduct cost management through tags.

> Note:
>
> Currently, the bucket tag feature supports a maximum of 50 different tags for a bucket.

Request

Request Example

```
PUT /?tagging HTTP 1.1
Host:<BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
Content-MD5: MD5
Content-Length: Content Length
Content-Type: application/xml

[Request Body]
```

> Note:
>
> Authorization: Auth String (For details, see Request Signature documentation).

Request Header

In addition to using common request headers, this API also supports the following request headers. For details about common request headers, see Public Request Headers documentation.

Name	Description	Type	Required
Content-MD5	The MD5 hash value of the request body content, encoded in Base64 as defined in RFC 1864, is used for integrity checking to verify whether the request body has changed during the transmission.	string	Yes

Request Body

This request needs to set the following tag set:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Tagging>
  <TagSet>
    <Tag>
      <Key>age</Key>
      <Value>18</Value>
    </Tag>
    <Tag>
      <Key>name</Key>
      <Value>xiaoming</Value>
    </Tag>
  </TagSet>
</Tagging>
```

The data are described as follows:

Node Name (Keyword)	Parent Node	Description	Type	Required
Tagging	No	Tag Set	Container	Yes
TagSet	Tagging	Tag Set	Container	Yes
Tag	Tagging.TagSet	Tag set, supports up to 10 tags.	Containers	Yes
Key	Tagging.TagSet.Tag	The length of the tag key is no more than 128 bytes, and the tag key supports English letters, numbers, spaces, plus signs, minus signs, underscores, equal signs, periods, colons, and slashes.	String	Yes
Value	Tagging.TagSet.Tag	The value of the tag is no longer than 256 bytes, and the value supports English letters, numbers, spaces, plus signs, minus signs, underscores, equal signs, periods, colons, and slashes.	String	Yes

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

The response body is empty.

Error Code

The following describes some special but common errors that may occur with this request:

Error Code	Description	HTTP Status Code
SignatureDoesNotMatch	If this error code is returned, it means the provided signature is incorrect.	403 Forbidden
NoSuchBucket	If this error code is returned, it means the bucket you want to add the rule to is not found.	404 Not Found
MalformedXML	The XML syntax of the request body is incorrect. Please compare it carefully with the restful API document.	400 Bad Request
BadRequest	If the maximum number of tags allowed for a bucket is exceeded, a maximum of 10 tags are currently supported.	400 Bad Request
InvalidTag	The key and value of the tag contain the reserved string cos: or Project.	400 Bad Request

Practical Case

Request

The following request writes two tags {age:18} and {name:xiaoming} to the bucket examplebucket-1250000000 . COS configures the tags successfully and returns a 204 success response.

```
PUT /?tagging HTTP/1.1
User-Agent: curl/7.29.0
Accept: */*
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=15
16361923;1517361973&q-key-time=1516361923;1517361973&q-url-param-list=tagging&q-header-lis
t=content-md5;host&q-signature=71251feb4501494edcfbd01747fa873003759404
Content-MD5: LIbd5t5HLPhuNWYkP6qHcQ==
```

```
Content-Length: 127
Content-Type: application/xml

<Tagging>
   <TagSet>
      <Tag>
         <Key>age</Key>
         <Value>18</Value>
      </Tag>
      <Tag>
         <Key>name</Key>
         <Value>xiaoming</Value>
      </Tag>
   </TagSet>
</Tagging>
```

Response

```
HTTP/1.1 204 No Content
Content-Type: application/xml
Content-Length: 0
Connection: keep-alive
Date: Fri, 19 Jan 2018 11:40:22 GMT
x-cos-request-id: NWE2MWQ5MjZfMTBhYzM1MGFfMTA5ODVfMTVjNDM=
```

Querying Bucket Tags

Description

COS allows you to query tags of an existing bucket. The GET Bucket tagging API is used to query existing bucket tags under a specified bucket.

> Note:
>
> If you are using a sub-account to call this API, please ensure that you have obtained permission for GET Bucket tagging from the root account.

Request

Request Example

```
GET /?tagging HTTP 1.1
Host:<BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (For details, see Request Signature documentation).

Request Header

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

The request body of this request is empty.

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

Upon successful query, returns application/xml data containing existing tag information for the bucket.

```
<Tagging>
   <TagSet>
      <Tag>
         <Key>string</Key>
         <Value>string</Value>
      </Tag>
      <Tag>
         <Key>string</Key>
         <Value>string</Value>
      </Tag>
   </TagSet>
</Tagging>
```

Specific node descriptions are as follows:

Node Name (Keyword)	Parent Node	Description	Type
Tagging	No	Tag Set	Container
TagSet	Tagging	Tag Set	Container
Tag	Tagging.TagSet	Tag Set, supporting up to 50 tags	Containers
Key	Tagging.TagSet.Tag	Tag Key, with a length not exceeding 128 bytes, supporting English letters, digits, spaces, plus signs, hyphens, underscores, equal signs, periods, colons, and slashes.	String
Value	Tagging.TagSet.Tag	Tag Value, with a length not exceeding 256 bytes, supporting English letters, digits, spaces, plus signs, hyphens, underscores, equal signs, periods, colons, and slashes.	String

Error Code

The following describes some special but common errors that may occur with this request:

Error Code	Description	HTTP Status Code
SignatureDoesNotMatch	If this error code is returned, it means the provided signature is incorrect.	403 Forbidden
NoSuchBucket	If this error code is returned, it means the bucket you want to add the rule to is not found.	404 Not Found

Error Code	Description	HTTP Status Code
NoSuchTagSetError	The bucket requested does not have any bucket tags set.	404 Not Found

Practical Case

Request

The following request applies to query tag information under the bucket examplebucket-1250000000 . COS parses this request and returns the existing tags {age:18} and {name:John Doe} under the bucket.

```
GET /?tagging HTTP/1.1
User-Agent: curl/7.29.0
Accept: */*
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=15
16361923;1517361973&q-key-time=1516361923;1517361973&q-url-param-list=tagging&q-header-lis
t=content-md5;host&q-signature=71251feb4501494edcfbd01747fa873003759404
Content-Length: 127
Content-Type: application/xml
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Connection: close
Date: Fri, 19 Jan 2018 11:40:22 GMT

<Tagging>
   <TagSet>
      <Tag>
         <Key>age</Key>
         <Value>18</Value>
      </Tag>
      <Tag>
         <Key>name</Key>
         <Value>John Doe</Value>
      </Tag>
   </TagSet>
</Tagging>
```

Deleting Bucket Tags

Description

COS allows you to delete tags for an existing bucket. The DELETE Bucket tagging API is used to delete existing bucket tags under a specified bucket.

> Note:
>
> If you are using a sub-account to call this API, please ensure that you have obtained permission for DELETE Bucket tagging from the root account.

Request

Request Example

```
DELETE /?tagging HTTP 1.1
Host:<BucketName-APPID>.<Endpoint>
Date:date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (For details, see Request Signature documentation).

Request Header

Common Headers

This request operation implementation uses common request headers. For details on common request headers, please refer to the Common Request Headers document.

Non-common Headers

This request has no special request header information.

Request Body

The request body of this request is empty.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details on common response headers, please refer to the Common Response Headers document.

Special Response Headers

This request has no special response header information.

Response Body

This request does not have any specific response body information.

Error Code

The following describes some special but common errors that may occur with this request:

Error Code	Description	HTTP Status Code
SignatureDoesNotMatch	If this error code is returned, it means the provided signature is incorrect.	403 Forbidden
NoSuchBucket	If this error code is returned, it means the bucket you want to add the rule to is not found.	404 Not Found
NoSuchTagSetError	The bucket requested does not have any bucket tags set.	404 Not Found

Practical Case

Request

The following request applies to delete the existing tag information under the bucket examplebucket-1250000000 . After parsing this request, COS deletes all tags under the bucket.

```
DELETE /?tagging HTTP/1.1
User-Agent: curl/7.29.0
Accept: */*
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=15
16361923;1517361973&q-key-time=1516361923;1517361973&q-url-param-list=tagging&q-header-lis
t=content-md5;host&q-signature=71251feb4501494edcfbd01747fa873003759404
Content-Md5: LIbd5t5HLPhuNWYkP6qHcQ==
Content-Length: 127
Content-Type: application/xml
```

Response

```
HTTP/1.1 204 No Content
Content-Type: application/xml
Connection: close
Date: Fri, 19 Jan 2018 11:40:22 GMT
```

Static Website (Website)
Setting a Static Website

Description

The PUT Bucket website request is used to configure a static website for a bucket. You can configure it by passing in an XML format configuration file, with a file size limit of 64 KB.
You are advised to debug this API through Platform Management > Cloud API Management > API Explorer in the console on the operation side.

> Note:
>
> API Explorer provides capabilities such as online calling, signature verification, SDK code generation, and quick interface search. You can view the request content and return results of each call and automatically generate SDK call examples.

Request

Request Example

```
PUT /?website HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Content-Type: application/xml
Content-Length: Content Length
Content-MD5: MD5
Authorization: Auth String

[Request Body]
```

> Note:
>
> - <BucketName-APPID> is the bucket name with APPID as the suffix, for example, examplebucket-1250000000.
> - Authorization: Auth String (For details, refer to Request Signature).

Request Parameters

This API has no request parameters.

Request Header

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

Submit application/xml request data containing the complete bucket static website configuration information.

```
<WebsiteConfiguration>
  <IndexDocument>
    <Suffix>String</Suffix>
  </IndexDocument>
  <RedirectAllRequestsTo>
    <Protocol>String</Protocol>
  </RedirectAllRequestsTo>
  <AutoAddressing>
    <Status>Enabled|Disabled</Status>
  </AutoAddressing>
  <ErrorDocument>
    <Key>String</Key>
    <OriginalHttpStatus>Enabled|Disabled</OriginalHttpStatus>
  </ErrorDocument>
  <RoutingRules>
    <RoutingRule>
      <Condition>
        <HttpErrorCodeReturnedEquals>Integer</HttpErrorCodeReturnedEquals>
      </Condition>
      <Redirect>
        <Protocol>String</Protocol>
        <ReplaceKeyWith>String</ReplaceKeyWith>
      </Redirect>
    </RoutingRule>
    <RoutingRule>
      <Condition>
        <KeyPrefixEquals>String</KeyPrefixEquals>
      </Condition>
      <Redirect>
        <Protocol>String</Protocol>
        <ReplaceKeyPrefixWith>String</ReplaceKeyPrefixWith>
      </Redirect>
    </RoutingRule>
  </RoutingRules>
</WebsiteConfiguration>
```

Specific node descriptions are as follows:

Node Name (Keyword)	Parent Node	Description	Type	Required
WebsiteConfiguration	No	Contains all request information for the PUT Bucket website operation.	Container	No

Container node WebsiteConfiguration contents:

Node Name (Keyword)	Parent Node	Description	Type	Required
IndexDocument	WebsiteConfiguration	Index document configuration	Container	Yes
RedirectAllRequestsTo	WebsiteConfiguration	Redirect all request configuration	Container	No
AutoAddressing	WebsiteConfiguration	Used to configure whether to ignore file extensions.	Container	No
ErrorDocument	WebsiteConfiguration	Error document configuration	Container	No
RoutingRules	WebsiteConfiguration	Redirect rule configuration, with a maximum of 100 RoutingRules set.	Container	No

Container node IndexDocument contents:

Node Name (Keyword)	Parent Node	Description	Type	Required
Suffix	WebsiteConfiguration.IndexDocument	Specify the object key suffix for the index document. For example, specifying index.html means that when accessing the root directory of the bucket, the content of index.html will be automatically returned, or when accessing the article/ directory, the content of article/ index.html will be automatically returned.	String	Yes

Container node RedirectAllRequestsTo contents:

Node Name (Keyword)	Parent Node	Description	Type	Required

Node Name (Keyword)	Parent Node	Description	Type	Required
Protocol	WebsiteConfiguration.RedirectAllRequestsTo	Specify the target protocol for redirecting all requests, which can only be set to HTTPS.	String	Yes

Container Node AutoAddressing Contents:

Node Name (Keyword)	Parent Node	Description	Type	Required
Status	WebsiteConfiguration.AutoAddressing	Used to configure whether to ignore HTML file extensions, with optional values of Enabled or Disabled, defaulting to Disabled.	String	No

Container node ErrorDocument contents:

Node Name (Keyword)	Parent Node	Description	Type	Required
Key	WebsiteConfiguration.ErrorDocument	Specify the object key for the common error document. When an error occurs and does not match the error code redirection in the redirect rule, the content of this object key will be returned.	String	Yes
OriginalHttpStatus	WebsiteConfiguration.ErrorDocument	Used to configure the HTTP status code that triggers the error document, with	String	No

Node Name (Keyword)	Parent Node	Description	Type	Required
		optional values of Enabled or Disabled, defaulting to Enabled.		

Container node RoutingRules contents:

Node Name (Keyword)	Parent Node	Description	Type	Required
RoutingRule	WebsiteConfiguration.RoutingRules	Single redirect rule configuration	Container	Yes

Container node RoutingRules.RoutingRule contents:

Node Name (Keyword)	Parent Node	Description	Type	Required
Condition	WebsiteConfiguration.RoutingRules.RoutingRule	Condition configuration for redirect rule	Container	Yes
Redirect	WebsiteConfiguration.RoutingRules.RoutingRule	Specific redirect target configuration for redirect rule	Container	Yes

Container node RoutingRules.RoutingRule.Condition contents:

Node Name (Keyword)	Parent Node	Description	Type	Required
HttpErrorCodeReturnedEquals	WebsiteConfiguration.RoutingRules.RoutingRule.Condition	Specify the error code matching condition for the redirect rule, supporting	Integer	HttpErrorCodeReturnedEquals and KeyPrefixEquals are mutually exclusive.

Node Name (Keyword)	Parent Node	Description	Type	Required
		only 4XX return codes, such as 403 or 404.		
KeyPrefixEquals	WebsiteConfiguration.RoutingRules.RoutingRule.Condition	Specify the object key prefix matching condition for the redirect rule.	String	HttpErrorCodeReturnedEquals and KeyPrefixEquals are mutually exclusive.

Container node RoutingRules.RoutingRule.Redirect contents:

Node Name (Keyword)	Parent Node	Description	Type	Required
Protocol	WebsiteConfiguration.RoutingRules.RoutingRule.Redirect	Specify the target protocol for the redirect rule, which can only be set to HTTPS.	String	No
ReplaceKeyWith	WebsiteConfiguration.RoutingRules.RoutingRule.Redirect	Specify the object key for the specific redirect target of the redirect rule, replacing the entire original request object key.	String	ReplaceKeyWith and ReplaceKeyPrefixWith are mutually exclusive.
ReplaceKeyPrefixWith	WebsiteConfiguration.RoutingRules.RoutingRule.Redirect	Specify the object key for the specific redirect target of the redirect rule, replacing the matched prefix portion of the original request. It can only be set when the Condition is KeyPrefixEquals.	String	ReplaceKeyWith and ReplaceKeyPrefixWith are mutually exclusive.

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

The response body for this API is empty.

Error Code

This API follows unified error response and error codes. For details, please refer to the Error Codes document.

Practical Case

Request

```
PUT /?website HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Wed, 20 May 2020 09:33:38 GMT
Content-Type: application/xml
Content-Length: 1209
Content-MD5: VHzj4Uwb++HLyCJp7jUzWg==
Authorization: q-sign-algorithm=sha1&q-ak=AKID8A0fBVtYFrNm02oY1g1JQQF0c3JO****&q-sign-time=1589967218;1589974418&q-key-time=1589967218;1589974418&q-header-list=content-length;content-md5;content-type;date;host&q-url-param-list=website&q-signature=4666493555640e834a879c78afaa4fd9b16a****
Connection: close

<WebsiteConfiguration>
  <IndexDocument>
    <Suffix>index.html</Suffix>
  </IndexDocument>
  <RedirectAllRequestsTo>
    <Protocol>https</Protocol>
  </RedirectAllRequestsTo>
  <ErrorDocument>
    <Key>pages/error.html</Key>
  </ErrorDocument>
  <RoutingRules>
    <RoutingRule>
      <Condition>
        <HttpErrorCodeReturnedEquals>403</HttpErrorCodeReturnedEquals>
      </Condition>
      <Redirect>
        <Protocol>https</Protocol>
        <ReplaceKeyWith>pages/403.html</ReplaceKeyWith>
```

```
        </Redirect>
      </RoutingRule>
      <RoutingRule>
        <Condition>
          <HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals>
        </Condition>
        <Redirect>
          <ReplaceKeyWith>pages/404.html</ReplaceKeyWith>
        </Redirect>
      </RoutingRule>
      <RoutingRule>
        <Condition>
          <KeyPrefixEquals>assets/</KeyPrefixEquals>
        </Condition>
        <Redirect>
          <ReplaceKeyWith>index.html</ReplaceKeyWith>
        </Redirect>
      </RoutingRule>
      <RoutingRule>
        <Condition>
          <KeyPrefixEquals>article/</KeyPrefixEquals>
        </Condition>
        <Redirect>
          <Protocol>https</Protocol>
          <ReplaceKeyPrefixWith>archived/</ReplaceKeyPrefixWith>
        </Redirect>
      </RoutingRule>
    </RoutingRules>
</WebsiteConfiguration>
```

Response

```
HTTP/1.1 200 OK
Content-Length: 0
Connection: close
Date: Wed, 20 May 2020 09:33:38 GMT
x-cos-request-id: NWVjNGY5NzJfOThjMjJhMDlfMjg5Ml8yYzNi****
```

Querying a Static Website

Description

The GET Bucket website request is used to query the static website configuration associated with a bucket. You are advised to debug this API through Platform Management > Cloud API Management > API Explorer in the console on the operation side.

> Note:
>
> API Explorer provides capabilities such as online calling, signature verification, SDK code generation, and quick interface search. You can view the request content and return results of each call and automatically generate SDK call examples.

Request

Request Example

```
GET /?website HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> - <BucketName-APPID> is the bucket name with APPID as the suffix, for example, examplebucket-1250000000.
> - Authorization: Auth String (For details, refer to Request Signature).

Request Parameters

This API has no request parameters.

Request Header

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

This API has no request body.

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

Upon successful query, it returns application/xml data, containing the complete bucket static website configuration information.

```
<WebsiteConfiguration>
  <IndexDocument>
    <Suffix>string</Suffix>
  </IndexDocument>
  <RedirectAllRequestsTo>
    <Protocol>string</Protocol>
  </RedirectAllRequestsTo>
  <ErrorDocument>
    <Key>string</Key>
  </ErrorDocument>
  <RoutingRules>
    <RoutingRule>
      <Condition>
        <HttpErrorCodeReturnedEquals>integer</HttpErrorCodeReturnedEquals>
      </Condition>
      <Redirect>
        <Protocol>string</Protocol>
        <ReplaceKeyWith>string</ReplaceKeyWith>
      </Redirect>
    </RoutingRule>
    <RoutingRule>
      <Condition>
        <KeyPrefixEquals>string</KeyPrefixEquals>
      </Condition>
      <Redirect>
        <Protocol>string</Protocol>
        <ReplaceKeyPrefixWith>string</ReplaceKeyPrefixWith>
      </Redirect>
    </RoutingRule>
  </RoutingRules>
</WebsiteConfiguration>
```

Specific node descriptions are as follows:

Node Name (Keyword)	Parent Node	Description	Type

Node Name (Keyword)	Parent Node	Description	Type
WebsiteConfiguration	No	Save all information from the GET Bucket website result.	Container

Container node WebsiteConfiguration contents:

Node Name (Keyword)	Parent Node	Description	Type
IndexDocument	WebsiteConfiguration	Index document configuration	Container
RedirectAllRequestsTo	WebsiteConfiguration	Redirect all request configuration	Container
ErrorDocument	WebsiteConfiguration	Error document configuration	Container
RoutingRules	WebsiteConfiguration	Redirect rule configuration.	Container

Container node IndexDocument contents:

Node Name (Keyword)	Parent Node	Description	Type
Suffix	WebsiteConfiguration.IndexDocument	Specify the object key suffix for the index document. For example, specifying index.html means that when accessing the root directory of the bucket, the content of index.html will be automatically returned, or when accessing the article/ directory, the content of article/index.html will be automatically returned.	string

Container node RedirectAllRequestsTo contents:

Node Name (Keyword)	Parent Node	Description	Type
Protocol	WebsiteConfiguration.RedirectAllRequestsTo	Specify the target protocol for redirecting all requests.	string

Container node ErrorDocument contents:

Node Name (Keyword)	Parent Node	Description	Type
Key	WebsiteConfiguration.ErrorDocument	Specify the object key for the common error document.	string

Container node RoutingRules contents:

Node Name (Keyword)	Parent Node	Description	Type
RoutingRule	WebsiteConfiguration.RoutingRules	Single redirect rule configuration	Container

Container node RoutingRules.RoutingRule contents:

Node Name (Keyword)	Parent Node	Description	Type
Condition	WebsiteConfiguration.RoutingRules.RoutingRule	Condition configuration for redirect rule	Container
Redirect	WebsiteConfiguration.RoutingRules.RoutingRule	Specific redirect target configuration for redirect rule	Container

Container node RoutingRules.RoutingRule.Condition contents:

Node Name (Keyword)	Parent Node	Description	Type
HttpErrorCodeReturnedEquals	WebsiteConfiguration.RoutingRules.RoutingRule.Condition	Specify the error code matching condition for the redirect rule.	integer
KeyPrefixEquals	WebsiteConfiguration.RoutingRules.RoutingRule.Condition	Specify the object key prefix matching condition for the redirect rule.	string

Container node RoutingRules.RoutingRule.Redirect contents:

Node Name	Parent Node	Description	Type

(Keyword)			
Protocol	WebsiteConfiguration.RoutingRules.RoutingRule.Redirect	Specify the target protocol for the redirect rule.	string
ReplaceKeyWith	WebsiteConfiguration.RoutingRules.RoutingRule.Redirect	Specify the object key for the specific redirect target of the redirect rule, replacing the entire original request object key.	string
ReplaceKeyPrefixWith	WebsiteConfiguration.RoutingRules.RoutingRule.Redirect	Specify the object key for the specific redirect target of the redirect rule, replacing the matched prefix portion of the original request.	string

Error Code

This API follows unified error response and error codes. For details, please refer to the Error Codes document.

Practical Case

Request

```
GET /?website HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Wed, 20 May 2020 09:33:49 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKID8A0fBVtYFrNm02oY1g1JQQF0c3JO****&q-sign-time=1589967229;1589974429&q-key-time=1589967229;1589974429&q-header-list=date;host&q-url-param-list=website&q-signature=50a22a30b02b59e5da4a0820d15a36805ea7****
Connection: close
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 1163
Connection: close
Date: Wed, 20 May 2020 09:33:49 GMT
x-cos-request-id: NWVjNGY5N2RfYTdjMjJhMDlfNjZkY18yYWUx****

<WebsiteConfiguration>
```

```
  <IndexDocument>
    <Suffix>index.html</Suffix>
  </IndexDocument>
  <RedirectAllRequestsTo>
    <Protocol>https</Protocol>
  </RedirectAllRequestsTo>
  <ErrorDocument>
    <Key>pages/error.html</Key>
  </ErrorDocument>
  <RoutingRules>
    <RoutingRule>
      <Condition>
        <HttpErrorCodeReturnedEquals>403</HttpErrorCodeReturnedEquals>
      </Condition>
      <Redirect>
        <Protocol>https</Protocol>
        <ReplaceKeyWith>pages/403.html</ReplaceKeyWith>
      </Redirect>
    </RoutingRule>
    <RoutingRule>
      <Condition>
        <HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals>
      </Condition>
      <Redirect>
        <ReplaceKeyWith>pages/404.html</ReplaceKeyWith>
      </Redirect>
    </RoutingRule>
    <RoutingRule>
      <Condition>
        <KeyPrefixEquals>assets/</KeyPrefixEquals>
      </Condition>
      <Redirect>
        <ReplaceKeyWith>index.html</ReplaceKeyWith>
      </Redirect>
    </RoutingRule>
    <RoutingRule>
      <Condition>
        <KeyPrefixEquals>article/</KeyPrefixEquals>
      </Condition>
      <Redirect>
        <Protocol>https</Protocol>
        <ReplaceKeyPrefixWith>archived/</ReplaceKeyPrefixWith>
      </Redirect>
    </RoutingRule>
  </RoutingRules>
</WebsiteConfiguration>
```

Deleting a Static Website

Description

The DELETE Bucket website request is used to delete the static website configuration in a bucket.
You are advised to debug this API through Platform Management > Cloud API Management > API Explorer in the console on the operation side.

> Note:
>
> API Explorer provides capabilities such as online calling, signature verification, SDK code generation, and quick interface search. You can view the request content and return results of each call and automatically generate SDK call examples.

Request

Request Example

```
DELETE /?website HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> - <BucketName-APPID> is the bucket name with APPID as the suffix, for example, examplebucket-1250000000.
> - Authorization: Auth String (For details, refer to Request Signature).

Request Parameters

This API has no request parameters.

Request Header

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

This API has no request body.

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

The response body for this API is empty.

Error Code

This API follows unified error response and error codes. For details, please refer to the Error Codes document.

Practical Case

Request

```
DELETE /?website HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Tue, 19 May 2020 07:57:10 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKID8A0fBVtYFrNm02oY1g1JQQF0c3JO****&q-sign-time=1589875030;1589882230&q-key-time=1589875030;1589882230&q-header-list=date;host&q-url-param-list=website&q-signature=e000543b192f0739b36f420456708fcfb553****
Connection: close
```

Response

```
HTTP/1.1 204 No Content
Connection: close
Date: Tue, 19 May 2020 07:57:10 GMT
x-cos-request-id: NWVjMzkxNTZfY2ZhZjJhMDlfNWI2OV8yYWFh****
Content-Length: 0
```

Inventory
Setting an Inventory Task

Description

The PUT Bucket inventory API is used to create an inventory task in a bucket. After naming your inventory task, you can use this request to create it. For details, see Inventory feature overview.

> Note
>
> - COS allows you to create up to 1000 inventory tasks in each bucket.
> - You should write a bucket policy to the destination bucket for COS to write the result files of the inventory tasks to the bucket.
> - To call this API, ensure that you have the necessary permission for inventory tasks in the bucket. The bucket owner has this permission by default. If you do not have it, you should apply for it from the bucket owner first.
> - If you specify a prefix for inventory delivery, the COS backend automatically adds / after the prefix you specified. If you specify Prefix as the prefix, the path of the inventory report delivered by the COS backend is Prefix/inventory_report .

Authorization Description

In the authorization policy, action is set to cos:PutBucketInventory.

Request

Request Example

```
PUT /?inventory&id=inventory-configuration-ID HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
Content-MD5: MD5
```

> Description
>
> Authorization: Auth String (For details, refer to Request Signature).

Request Parameters

To call the PUT Bucket inventory API, the inventory task name needs to be used. The format of this parameter is as follows:

Parameter	Description	Type	Required
id	Inventory task name. Default value: None Valid characters: a–z, A–Z, 0–9, -, _, .	String	Yes

Request Header

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

Set the specific configuration of the inventory task in the request body by using XML. The configuration includes the analysis objects, analysis frequency, analysis dimensions, and analysis result format and storage path of the inventory task.

```
<InventoryConfiguration>
  <Id>list1</Id>
  <IsEnabled>true</IsEnabled>
  <Destination>
    <COSBucketDestination>
      <Format>CSV</Format>
      <AccountId>100000000001</AccountId>
      <Bucket>qcs::cos:region1::examplebucket-1250000000</Bucket>
      <Prefix>list1</Prefix>
      <Encryption>
        <SSE-COS></SSE-COS>
      </Encryption>
    </COSBucketDestination>
  </Destination>
  <Schedule>
    <Frequency>Daily|Weekly|Monthly</Frequency>
    <MonthlyDate></MonthlyDate>
  </Schedule>
  <Filter>
    <Prefix>myPrefix</Prefix>
  </Filter>
  <IncludedObjectVersions>All</IncludedObjectVersions>
  <OptionalFields>
    <Field>Size</Field>
    <Field>LastModifiedDate</Field>
    <Field>ETag</Field>
    <Field>StorageClass</Field>
    <Field>IsMultipartUploaded</Field>
    <Field>ReplicationStatus</Field>
```

```
        <Field>Tag</Field>
        <Field>Crc64</Field>
        <Field>x-cos-meta-*</Field>
    </OptionalFields>
</InventoryConfiguration>
```

Specific node descriptions are as follows:

Node Name	Parent Node	Description	Type	Required
InventoryConfiguration	No	Inventory configuration parameters.	Container	Yes
Id	InventoryConfiguration	Name of the inventory, corresponding to the id parameter in the request.	Container	Yes
IsEnabled	InventoryConfiguration	Identifier of whether the inventory is enabled. - If it is set to true, the inventory feature takes effect. - If it is set to false, no inventory is generated.	String	Yes
IncludedObjectVersions	InventoryConfiguration	Whether to include the object version in the inventory. - If it is set to All, the inventory includes all object versions, and the VersionId, IsLatest, and DeleteMarker fields are added to the inventory. - If it is set to Current, the inventory does not include the object version information.	String	Yes
Filter	InventoryConfiguration	Filters out objects for analysis. The inventory feature	Container	No

Node Name	Parent Node	Description	Type	Required
		analyzes objects that match the prefix set in Filter.		
And	Filter	If both the prefix and object tag conditions are required for filtering out objects to be analyzed, And is required.	Container	No
Prefix	And	Prefix of the object to be analyzed.	String	No
Tag	And	When objects for analysis are filtered out, one or more object tags can be used as filter conditions.	Container	No
Period	Filter	Creation time range of the objects to be analyzed.	Container	No
StartTime	Period	Start creation time of the objects to be analyzed. The parameter is a second-level timestamp, such as 1568688761.	String	No
EndTime	Period	End creation time of the objects to be analyzed. The parameter is a second-level timestamp, such as 1568688762.	String	No
OptionalFields	InventoryConfiguration	Analysis items that should be included in the inventory result.	Container	No
Field	OptionalFields	Names of optional analysis items in the	String	No

Node Name	Parent Node	Description	Type	Required
		inventory result. Optional fields include: Size , LastModifiedDate , StorageClass , ETag , IsMultipartUploaded , ReplicationStatus , Tag , Crc64 , and x-cos-meta-* . Note: If object tags are used in the filter conditions, tags should be added here. In addition, users can enter custom headers in x-cos-meta-* format, such as x-cos-meta-testheader . The inventory outputs the corresponding object metadata. If the object does not contain that metadata, it is empty.		
Schedule	InventoryConfiguration	Configures the inventory task period.	Container	Yes
Frequency	Schedule	Inventory task period. Options include Daily, Weekly, or Monthly. Enumerated values: Daily, Weekly. If Monthly is specified, the specific inventory output date needs to be specified.	String	Yes
MonthlyDate	Schedule	If Monthly is specified, the specific inventory output date needs to be specified. The value range is 1 to 28.	Integer	No
Destination	InventoryConfiguration	Information about the inventory result	Container	Yes

Node Name	Parent Node	Description	Type	Required
		destination.		
COSBucketDestination	Destination	Information about the bucket where the exported inventory results are stored.	Container	Yes
Bucket	COSBucketDestination	Name of the bucket where the inventory analysis results are stored.	String	Yes
AccountId	COSBucketDestination	UIN (root account) of the bucket owner, for example, 100000000001.	String	No
Prefix	COSBucketDestination	Prefix of the inventory analysis results.	String	No
Format	COSBucketDestination	File format of the inventory analysis results. An option is the CSV format.	String	Yes
Encryption	COSBucketDestination	Option for SSE of the inventory results.	Container	No
SSE-COS	Encryption	Encryption method of the COS-managed key, which does not need to be specified.	Container	No

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

The response body for this request is empty.

Error Code

This API follows unified error response and error codes. For details, please refer to the Error Codes document.

Practical Case

Example 1: Adding an Inventory Task with an Object Prefix Specified

Request

This example adds the inventory task list1 to the bucket examplebucket-1250000000 .

- The inventory task analyzes objects with the prefix myPrefix in the bucket and their versions.
- The analysis frequency is once per day.
- Analysis dimensions include Size, LastModifiedDate, StorageClass, ETag, IsMultipartUploaded, and ReplicationStatus.
- The analysis results are stored as a CSV file in the bucket examplebucket-1250000000 . The file has the prefix list1 and is encrypted by using SSE-COS.

```
PUT /?inventory&id=list1 HTTP/1.1
Date: Mon, 28 Aug 2018 02:53:38 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDZfbOAo7cllgPvF9cXFrJD0a1ICvR****&q-sign-time=1
503888878;1503889238&q-key-time=1503888878;1503889238&q-header-list=host&q-url-param-list=
inventory&q-signature=254bf9cd3d6615e89a36ab652437f9d45c5f****
Content-MD5: AAq9nzrpsz5LJ4UEe1f6Q==
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Content-Length: 1024

<?xml version = "1.0" encoding = "UTF-8">
<InventoryConfiguration xmlns = "http://....">
   <Id>list1</Id>
   <IsEnabled>true</IsEnabled>
   <Destination>
      <COSBucketDestination>
         <Format>CSV</Format>
         <AccountId>100000000001</AccountId>
         <Bucket>qcs::cos:region1::examplebucket-1250000000</Bucket>
         <Prefix>list1</Prefix>
         <Encryption>
            <SSE-COS></SSE-COS>
         </Encryption>
      </COSBucketDestination>
   </Destination>
   <Schedule>
      <Frequency>Daily</Frequency>
```

```
    </Schedule>
    <Filter>
      <Prefix>myPrefix</Prefix>
     <Period>
      <StartTime>1568688761</StartTime>
      <EndTime>1568688762</EndTime>
     </Period>
    </Filter>
    <IncludedObjectVersions>All</IncludedObjectVersions>
    <OptionalFields>
      <Field>Size</Field>
      <Field>LastModifiedDate</Field>
      <Field>ETag</Field>
      <Field>StorageClass</Field>
      <Field>IsMultipartUploaded</Field>
      <Field>ReplicationStatus</Field>
    </OptionalFields>
</InventoryConfiguration>
```

Response

After receiving the preceding request, COS returns the following response, indicating that the inventory task list1 is set successfully.

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Date: Mon, 28 Aug 2018 02:53:38 GMT
x-cos-request-id: NTlhMzg1ZWVfMjQ4OGY3MGFfMWE1NF8****
```

Example 2: Adding an Inventory Task with an Object Prefix and Object Tag Specified

Request

This example adds the inventory task list2 to the bucket examplebucket-1250000000 .

- The inventory task analyzes objects with the prefix myPrefix and object tag {age:18} in the bucket and all their versions.
- The analysis frequency is once per day.
- Analysis dimensions include Size, LastModifiedDate, StorageClass, ETag, and Tag.
- The analysis results are stored as a CSV file in the bucket inventorybucket-1250000000 .

```
PUT /?inventory&id=list2 HTTP/1.1
Date: Mon, 28 Aug 2018 02:53:38 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDZfbOAo7cllgPvF9cXFrJD0a1ICvR****&q-sign-time=1
```

```
503888878;1503889238&q-key-time=1503888878;1503889238&q-header-list=host&q-url-param-list=
inventory&q-signature=254bf9cd3d6615e89a36ab652437f9d45c5f****
Content-MD5: AAq9nzrpsz5LJ4UEe1f6Q==
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Content-Length: 1024

<?xml version = "1.0" encoding = "UTF-8">
<InventoryConfiguration xmlns = "http://....">
   <Id>list2</Id>
   <IsEnabled>true</IsEnabled>
   <Destination>
      <COSBucketDestination>
         <Format>CSV</Format>
         <AccountId>100000000001</AccountId>
         <Bucket>qcs::cos:region1::inventorybucket-1250000000</Bucket>
      </COSBucketDestination>
   </Destination>
   <Schedule>
      <Frequency>Daily</Frequency>
   </Schedule>
   <Filter>
      <And>
         <Prefix>myPrefix</Prefix>
         <Tag>
            <Key>age</Key>
            <Value>18</Value>
         </Tag>
      </And>
   </Filter>
   <IncludedObjectVersions>All</IncludedObjectVersions>
   <OptionalFields>
      <Field>Size</Field>
      <Field>LastModifiedDate</Field>
      <Field>StorageClass</Field>
      <Field>ETag</Field>
      <Field>Tag</Field>
   </OptionalFields>
</InventoryConfiguration>
```

Response

After receiving the preceding request, COS returns the following response, indicating that the inventory task list2 is set successfully.

```
HTTP/1.1 200 OK
Content-Type: application/xml
```

```
Content-Length: 0
Date: Mon, 28 Aug 2018 02:53:38 GMT
x-cos-request-id: NTlhMzg1ZWVfMjQ4OGY3MGFfMWE1NF8****
```

Example 3: Adding an Inventory Task to Export Custom Headers of an Object

Request

This example adds the inventory task list3 to the bucket examplebucket-1250000000 .

- The inventory task analyzes objects with the prefix myPrefix and object tag {age:18} in the bucket and all their versions.
- The analysis frequency is once per day.
- Analysis dimensions include Size, LastModifiedDate, StorageClass, ETag, Tag, and x-cos-meta-myheader.
- The analysis results are stored as a CSV file in the bucket inventorybucket-1250000000 .

```
PUT /?inventory&id=list2 HTTP/1.1
Date: Mon, 28 Aug 2018 02:53:38 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDZfbOAo7cllgPvF9cXFrJD0a1ICvR****&q-sign-time=1
503888878;1503889238&q-key-time=1503888878;1503889238&q-header-list=host&q-url-param-list=
inventory&q-signature=254bf9cd3d6615e89a36ab652437f9d45c5f****
Content-MD5: AAq9nzrpsz5LJ4UEe1f6Q==
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Content-Length: 1024

<?xml version = "1.0" encoding = "UTF-8">
<InventoryConfiguration xmlns = "http://....">
   <Id>list3</Id>
   <IsEnabled>true</IsEnabled>
   <Destination>
      <COSBucketDestination>
         <Format>CSV</Format>
         <AccountId>100000000001</AccountId>
         <Bucket>qcs::cos:region1::inventorybucket-1250000000</Bucket>
      </COSBucketDestination>
   </Destination>
   <Schedule>
      <Frequency>Daily</Frequency>
   </Schedule>
   <Filter>
      <And>
         <Prefix>myPrefix</Prefix>
         <Tag>
            <Key>age</Key>
```

```
            <Value>18</Value>
          </Tag>
        </And>
      </Filter>
      <IncludedObjectVersions>All</IncludedObjectVersions>
      <OptionalFields>
        <Field>Size</Field>
        <Field>LastModifiedDate</Field>
        <Field>StorageClass</Field>
        <Field>ETag</Field>
        <Field>Tag</Field>
        <Field>Crc64</Field>
        <Field>x-cos-meta-myheader</Field>
      </OptionalFields>
    </InventoryConfiguration>
```

Response

After receiving the preceding request, COS returns the following response, indicating that the inventory task list3 is set successfully.

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Date: Mon, 28 Aug 2018 02:53:38 GMT
x-cos-request-id: NTlhMzg1ZWVfMjQ4OGY3MGFfMWE1NF8****
```

Creating a One-time Inventory Task

Description

The POST Bucket inventory API is used to create a one-time inventory task for a bucket. Unlike an inventory task created by using the PUT Bucket inventory API, an inventory task created by using this API starts immediately after creation, and each task is executed only once instead of being executed periodically. This feature allows you to flexibly obtain the list of objects in the bucket, ensuring more refined object management.

> Note
>
> - You should write a bucket policy to the destination bucket for COS to write the result files of the inventory tasks to the bucket.
> - To call this API, ensure that you have the necessary permission for inventory tasks in the bucket. The bucket owner has this permission by default. If you do not have it, you should apply for it from the bucket owner first.
> - If you specify a prefix for inventory delivery, the COS backend automatically adds `/` after the prefix you specified. For example, if you specify `Prefix` as the prefix, the path of inventory report delivered by the COS backend is `Prefix/inventory_report` .
> - If you submit another inventory task with the same ID as the ongoing inventory task or a periodic inventory task, the server returns a duplicate inventory task error.

Request

Request Example

```
POST /?inventory&id=inventory-id HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
Content-MD5: MD5
```

> Description
>
> Authorization: Auth String (For details, see Request Signature.)

Request Parameters

To call the POST Bucket inventory API, the inventory task name is required. The format of this parameter is as follows:

Parameter	Description	Type	Required
id	Inventory task name. Default value: None Valid characters: a–z, A–Z, 0–9, -, _, . Note: It is recommended that the inventory task ID not be the same as that of a periodic inventory task. Submitting inventory tasks with the same ID is not allowed within a day. Otherwise, an error is returned.	String	Yes

Request Header

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

Set the specific configuration of the inventory task using XML in the request body. An example is as follows:

> Note
>
> - The one-time inventory job cannot include IsEnabled or Schedule parameter.
> - Other parameters are the same as those in PUT Bucket inventory.
> - The storage path of the one-time inventory report is the same as that of the periodic inventory report. For details, see Inventory Report Storage Path.

```
<InventoryConfiguration>
  <Id>inventory-id</Id>
  <Destination>
    <COSBucketDestination>
      <Format>CSV</Format>
      <AccountId>100000000001</AccountId>
      <Bucket>qcs::cos:city::examplebucket-1250000000</Bucket>
      <Prefix>cos_bucket_inventory</Prefix>
      <Encryption>
        <SSE-COS></SSE-COS>
      </Encryption>
    </COSBucketDestination>
  </Destination>
  <Filter>
    <And>
      <Prefix>myPrefix</Prefix>
      <Tag>
        <Key>string</Key>
        <Value>string</Value>
      </Tag>
    </And>
  </Filter>
```

```
    <IncludedObjectVersions>All</IncludedObjectVersions>
    <OptionalFields>
      <Field>Size</Field>
      <Field>LastModifiedDate</Field>
      <Field>ETag</Field>
      <Field>StorageClass</Field>
      <Field>IsMultipartUploaded</Field>
      <Field>ReplicationStatus</Field>
      <Field>Tag</Field>
      <Field>Crc64</Field>
      <Field>x-cos-meta-*</Field>
    </OptionalFields>
</InventoryConfiguration>
```

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

The response body for this request is empty.

Error Code

This API follows unified error response and error codes. For details, please refer to the Error Codes document.

Practical Case

Example: Filtering by the Specified Prefix to Generate a One-time Inventory

Request

This example initiates the inventory task disposable in the bucket examplebucket-1250000000 .

- The inventory task analyzes objects with the prefix myPrefix and object tag {age:18} in the bucket and their versions.
- The analysis dimensions include Size, LastModifiedDate, StorageClass, ETag, and Tag.
- The analysis results are stored as a CSV file in the bucket inventorybucket-1250000000.
- If the inventory task was initiated on November 20, 2022, Manifest-related files of the inventory were placed in the following path in the destination bucket.

cos_bucket_inventory/1250000000/examplebucket/disposable/20221120/manifest.json
cos_bucket_inventory/1250000000/examplebucket/disposable/20221120/manifest.checksum

```
POST /?inventory&id=disposable HTTP/1.1
Date: Mon, 28 Aug 2018 02:53:38 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDZfbOAo7cllgPvF9cXFrJD0a1ICvR****&q-sign-time=
1503888878;1503889238&q-key-time=1503888878;1503889238&q-header-list=host&q-url-param-list
=inventory&q-signature=254bf9cd3d6615e89a36ab652437f9d45c5f****
Content-MD5: AAq9nzrpsz5LJ4UEe1f6Q==
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Content-Length: 1024

<?xml version = "1.0" encoding = "UTF-8">
<InventoryConfiguration xmlns = "http://....">
   <Id>disposable</Id>
   <Destination>
      <COSBucketDestination>
         <Format>CSV</Format>
         <AccountId>100000000001</AccountId>
         <Bucket>qcs::cos:city::inventorybucket-1250000000</Bucket>
         <Prefix>cos_bucket_inventory</Prefix>
      </COSBucketDestination>
   </Destination>
   <Filter>
      <And>
         <Prefix>myPrefix</Prefix>
         <Tag>
            <Key>age</Key>
            <Value>18</Value>
         </Tag>
      </And>
   </Filter>
   <IncludedObjectVersions>All</IncludedObjectVersions>
   <OptionalFields>
      <Field>Size</Field>
      <Field>LastModifiedDate</Field>
      <Field>StorageClass</Field>
      <Field>ETag</Field>
      <Field>Tag</Field>
   </OptionalFields>
</InventoryConfiguration>
```

Response

After receiving the preceding request, COS returns the following response, indicating that the inventory

task is submitted successfully.

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Date: Mon, 28 Aug 2018 02:53:38 GMT
x-cos-request-id: NTlhMzg1ZWVfMjQ4OGY3MGFfMWE1NF8****
```

Querying an Inventory Task

Description

The GET Bucket inventory API is used to query the inventory task information of a user in a bucket. When initiating this request, the user should provide the inventory task name and obtain a request signature to indicate that the request is authorized. For detailed inventory features, see Inventory feature overview.

> Note
>
> - To call this API, ensure that you have the necessary permission for inventory tasks in the bucket.
> - The bucket owner has this permission by default. If you do not have it, you should apply for it from the bucket owner first.
> - If you specify a prefix for inventory delivery, the COS backend automatically adds / after the prefix you specified. If you specify Prefix as the prefix, the path of the inventory report delivered by the COS backend is Prefix/inventory_report .

Authorization Description

In the authorization policy, action is set to cos:GetBucketInventory.

Request

Request Example

```
GET /?inventory&id=inventory-configuration-ID HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Description
>
> Authorization: Auth String (For details, refer to Request Signature).

Request Parameters

To call the GET Bucket inventory API, the inventory task name is required. The format of this parameter is as follows:

Parameter	Description	Type	Required

Parameter	Description	Type	Required
id	Inventory task name. Default value: None Valid characters: a–z, A–Z, 0–9, -, _, .	String	Yes

Request Header

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

The request body of this request is empty.

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

The response body returns application/xml data. The following contains all the nodes:

```
<InventoryConfiguration>
  <Id>list1</Id>
  <IsEnabled>true</IsEnabled>
  <Destination>
    <COSBucketDestination>
      <Format>CSV</Format>
      <AccountId>1250000000</AccountId>
      <Bucket>qcs::cos:city::examplebucket-1250000000</Bucket>
      <Prefix>list1</Prefix>
      <Encryption>
        <SSE-COS></SSE-COS>
      </Encryption>
    </COSBucketDestination>
  </Destination>
  <Schedule>
    <Frequency>Daily</Frequency>
  </Schedule>
  <Filter>
    <Prefix>myPrefix</Prefix>
  </Filter>
  <IncludedObjectVersions>All</IncludedObjectVersions>
  <OptionalFields>
```

```
        <Field>Size</Field>
        <Field>LastModifiedDate</Field>
        <Field>ETag</Field>
        <Field>StorageClass</Field>
        <Field>IsMultipartUploaded</Field>
        <Field>ReplicationStatus</Field>
        <Field>Tag</Field>
        <Field>Crc64</Field>
        <Field>x-cos-meta-*</Field>
    </OptionalFields>
</InventoryConfiguration>
```

Specific node descriptions are as follows:

Node Name	Parent Node	Description	Type
InventoryConfiguration	No	Inventory configuration parameters.	Container
Id	InventoryConfiguration	Name of the inventory, corresponding to the id parameter in the request.	Container
IsEnabled	InventoryConfiguration	Identifier of whether the inventory is enabled. - If it is set to true, the inventory feature takes effect. - If it is set to false, no inventory is generated.	String
IncludedObjectVersions	InventoryConfiguration	Whether to include the object version in the inventory. - If it is set to All, the inventory includes all object versions, and the VersionId, IsLatest, and DeleteMarker fields are added to the inventory. - If it is set to Current, the inventory does not include the object version information.	String
Filter	InventoryConfiguration	Filters out objects for analysis. The inventory feature analyzes objects that match the prefix set in Filter.	Container
And	Filter	If both the prefix and object tag conditions are required for filtering out objects to be	Container

Node Name	Parent Node	Description	Type
		analyzed, And is required.	
Prefix	And	Prefix of the object to be analyzed.	String
Tag	And	When objects for analysis are filtered out, one or more object tags can be used as filter conditions.	Container
Period	Filter	Creation time range of the objects to be analyzed.	Container
StartTime	Period	Start creation time of the objects to be analyzed. The parameter is a second-level timestamp, such as 1568688761.	String
EndTime	Period	End creation time of the objects to be analyzed. The parameter is a second-level timestamp, such as 1568688762.	String
OptionalFields	InventoryConfiguration	Analysis dimensions that should be included in the inventory results.	Container
Field	OptionalFields	Names of optional analysis items in the inventory result. Optional fields include: Size , LastModifiedDate , StorageClass , ETag , IsMultipartUploaded , ReplicationStatus , Tag , Crc64 , and x-cos-meta-* . Note: If object tags are used in the filter conditions, tags should be added here. In addition, users can enter custom headers in x-cos-meta-* format, such as x-cos-meta-testheader . The inventory outputs the corresponding object metadata. If the object does not contain that metadata, it is empty.	String
Schedule	InventoryConfiguration	Configures the inventory task period.	Container

Node Name	Parent Node	Description	Type
Frequency	Schedule	Inventory task period. Options include By day and By week.	String
Destination	InventoryConfiguration	Information about the inventory result destination.	Container
COSBucketDestination	Destination	Information about the bucket where the exported inventory results are stored.	Container
Bucket	COSBucketDestination	Name of the bucket where the inventory analysis results are stored.	String
AccountId	COSBucketDestination	ID of the bucket owner.	String
Prefix	COSBucketDestination	Prefix of the inventory analysis results.	String
Format	COSBucketDestination	File format of the inventory analysis results. Options include CSV and ORC.	String
Encryption	COSBucketDestination	Option for SSE of the inventory results.	Container
SSE-COS	Encryption	Encryption method of the COS-managed key.	Container

Error Code

This API follows unified error response and error codes. For details, please refer to the Error Codes document.

Practical Case

Example 1: Obtaining the Configuration of an Inventory Task for Obtaining Objects with a Specified Object Prefix

Request

The following example queries the list1 inventory job configuration of the examplebucket-1250000000 .

```
GET /?inventory&id=list1 HTTP/1.1
Date: Mon, 28 Aug 2018 02:53:38 GMT
```

```
Authorization: q-sign-algorithm=sha1&q-ak=AKIDZfbOAo7cllgPvF9cXFrJD0a1ICvR****&q-sign-time=
1503895278;1503895638&q-key-time=1503895278;1503895638&q-header-list=host&q-url-param-list
=inventory&q-signature=f77900be432072b16afd8222b4b349aabd83****
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
```

Response

After receiving the request, COS returns the following response, indicating that the inventory task list1 in the bucket is enabled.

- The inventory task analyzes objects with the prefix myPrefix in the bucket examplebucket-1250000000 and all their versions.
- The analysis frequency is once per day.
- Analysis dimensions include Size, LastModifiedDate, StorageClass, ETag, IsMultipartUploaded, and ReplicationStatus.
- Analysis results are stored as a CSV file in the bucket examplebucket-1250000000. The file has a prefix list1 and is encrypted by using SSE-COS.

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 331
Date: Mon, 28 Aug 2018 02:53:39 GMT
x-cos-request-id: NTlhMzg1ZWVfMjQ4OGY3MGFfMWE1NF84Y2M
<?xml version = "1.0" encoding = "UTF-8">
<InventoryConfiguration xmlns = "http://....">
   <Id>list1</Id>
   <IsEnabled>true</IsEnabled>
   <Destination>
      <COSBucketDestination>
         <Format>CSV</Format>
         <AccountId>1250000000</AccountId>
         <Bucket>qcs::cos:city::examplebucket-1250000000</Bucket>
         <Prefix>list1</Prefix>
         <Encryption>
            <SSE-COS></SSE-COS>
         </Encryption>
      </COSBucketDestination>
   </Destination>
   <Schedule>
      <Frequency>Daily</Frequency>
   </Schedule>
   <Filter>
      <Prefix>myPrefix</Prefix>
   </Filter>
   <IncludedObjectVersions>All</IncludedObjectVersions>
```

```
    <OptionalFields>
      <Field>Size</Field>
      <Field>LastModifiedDate</Field>
      <Field>ETag</Field>
      <Field>StorageClass</Field>
      <Field>IsMultipartUploaded</Field>
      <Field>ReplicationStatus</Field>
    </OptionalFields>
</InventoryConfiguration>
```

Example 2: Obtaining the Configuration of an Inventory Task for Obtaining Objects with a Specified Object Prefix and Object Tag

Request

The following example queries the list2 inventory job configuration of the examplebucket-1250000000 .

```
GET /?inventory&id=list2 HTTP/1.1
Date: Mon, 28 Aug 2018 02:53:38 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDZfbOAo7cllgPvF9cXFrJD0a1ICvR****&q-sign-time=
1503895278;1503895638&q-key-time=1503895278;1503895638&q-header-list=host&q-url-param-list
=inventory&q-signature=f77900be432072b16afd8222b4b349aabd83****
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
```

Response

After receiving the request, COS returns the following response, indicating that the inventory task list2 in the bucket is enabled.

- The inventory task analyzes objects with the prefix myPrefix, and object tag {age:18} in the bucket examplebucket-1250000000 and all their versions.
- The analysis frequency is once per day.
- Analysis dimensions include Size, LastModifiedDate, StorageClass, ETag, and Tag.
- The analysis results are stored as a CSV file in the bucket inventorybucket-1250000000.

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 331
Date: Mon, 28 Aug 2018 02:53:39 GMT

x-cos-request-id: NTlhMzg1ZWVfMjQ4OGY3MGFfMWE1NF84Y2M
<?xml version = "1.0" encoding = "UTF-8">
<InventoryConfiguration xmlns = "http://....">
    <Id>list2</Id>
    <IsEnabled>true</IsEnabled>
```

```
    <Destination>
      <COSBucketDestination>
        <Format>CSV</Format>
        <AccountId>1250000000</AccountId>
        <Bucket>qcs::cos:city::inventorybucket-1250000000</Bucket>
      </COSBucketDestination>
    </Destination>
    <Schedule>
      <Frequency>Daily</Frequency>
    </Schedule>
    <Filter>
      <And>
        <Prefix>myPrefix</Prefix>
        <Tag>
          <Key>age</Key>
          <Value>18</Value>
        </Tag>
      </And>
    </Filter>
    <IncludedObjectVersions>All</IncludedObjectVersions>
    <OptionalFields>
      <Field>Size</Field>
      <Field>LastModifiedDate</Field>
      <Field>StorageClass</Field>
      <Field>ETag</Field>
      <Field>Tag</Field>
    </OptionalFields>
</InventoryConfiguration>
```

Querying All Inventory Tasks

Description

The List Bucket Inventory Configurations API is used to query all inventory tasks in a bucket. You can configure up to 1,000 inventory tasks in a bucket.
This request supports pagination, returning up to 100 inventory tasks per page. Please check the value of the IsTruncated node in the request.

- If IsTruncated is false, all inventory tasks in the bucket are listed.
- If IsTruncated is true and there is a parameter value in the NextContinuationToken node, you can pass the value of the NextContinuationToken node to the continuation-token node to obtain information about inventory tasks on the next page.
  For detailed inventory features, see Inventory Feature Overview.

  > Note
  >
  > To call this API, ensure that you have the necessary permission for inventory tasks in the bucket. The bucket owner has this permission by default. If you do not have it, you should apply for it from the bucket owner first.

Authorization Description

In the authorization policy, action is set to cos:GetBucketInventory.

Request

Request Example

```
GET /?inventory HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Description
>
> Authorization: Auth String (For details, refer to Request Signature).

Request Parameters

The following table describes the request parameter format.

Parameter	Description	Type	Required
continuation-token	If IsTruncated in the COS response body is true and there is a parameter value in the NextContinuationToken node, you can use this parameter as the value of the continuation-token node to obtain information about inventory tasks on the next page. Default value: None	String	No

Request Header

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

The request body of this request is empty.

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

The response body returns application/xml data. The following contains all the nodes:

```
<ListInventoryConfigurationResult>
  <InventoryConfiguration>
    <Id>list1</Id>
    <IsEnabled>True</IsEnabled>
    <Destination>
      <COSBucketDestination>
        <Format>CSV</Format>
        <AccountId>1250000000</AccountId>
        <Bucket>qcs::cos:city::examplebucket-1250000000</Bucket>
        <Prefix>list1</Prefix>
        <SSE-COS></SSE-COS>
      </COSBucketDestination>
    </Destination>
    <Schedule>
      <Frequency>Daily</Frequency>
    </Schedule>
    <Filter>
      <Prefix>myPrefix</Prefix>
```

```
        </Filter>
        <IncludedObjectVersions>All</IncludedObjectVersions>
        <OptionalFields>
            <Field>Size</Field>
            <Field>LastModifiedDate</Field>
            <Field>ETag</Field>
            <Field>StorageClass</Field>
            <Field>IsMultipartUpload</Field>
            <Field>ReplicationStatus</Field>
        </OptionalFields>
    </InventoryConfiguration>
    <InventoryConfiguration>
        <Id>list2</Id>
        <IsEnabled>True</IsEnabled>
        <Destination>
            <COSBucketDestination>
                <Format>CSV</Format>
                <AccountId>1250000000</AccountId>
                <Bucket>qcs::cos:city::examplebucket-1250000000</Bucket>
                <Prefix>list2</Prefix>
                <SSE-COS></SSE-COS>
            </COSBucketDestination>
        </Destination>
        <Schedule>
            <Frequency>Weekly</Frequency>
        </Schedule>
        <Filter>
            <Prefix>myPrefix2</Prefix>
        </Filter>
        <IncludedObjectVersions>All</IncludedObjectVersions>
        <OptionalFields>
            <Field>Size</Field>
            <Field>LastModifiedDate</Field>
            <Field>ETag</Field>
            <Field>StorageClass</Field>
        </OptionalFields>
    </InventoryConfiguration>
    <IsTruncated>false</IsTruncated>
    ------If ContinuationToken was provided in the request---
    <ContinuationToken>...</ContinuationToken>
    <IsTruncated>true</IsTruncated>
    <NextContinuationToken>1ueSDFASDF1Tr/XDAFdadEADadf2J/wm36Hy4vbOwM=</NextContinuati
onToken>
</ListInventoryConfigurationResult>
```

Specific node descriptions are as follows:

Node Name	Parent Node	Description	Type
ListInventoryConfigurationResult	No	Information about all inventory jobs of the bucket.	Container
InventoryConfiguration	ListInventoryConfigurationResult	Detailed information about the inventory task. For its XML structure, see GET Bucket Inventory.	Container
IsTruncated	ListInventoryConfigurationResult	Indicates whether all inventory task information is listed. If all inventory task information is displayed, it is false. Otherwise, it is true.	Boolean
ContinuationToken	ListInventoryConfigurationResult	Identifier of the current inventory list page, which can be understood as the page number. This identifier corresponds to the continuation-token parameter in the request.	String
NextContinuationToken	ListInventoryConfigurationResult	Identifier of the next inventory list page. If this parameter has a	String

Node Name	Parent Node	Description	Type
		value, you can use this value as the continuation-token parameter and send a GET request to obtain information about inventory tasks on the next page.	

Error Code

This API follows unified error response and error codes. For details, please refer to the Error Codes document.

Practical Case

Request

The following example queries the list1 inventory job configuration of the examplebucket-1250000000 bucket

```
GET /?inventory HTTP/1.1
Date: Mon, 28 Aug 2018 02:53:38 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=15
03895278;1503895638&q-key-time=1503895278;1503895638&q-header-list=host&q-url-param-list=i
nventory&q-signature=f77900be432072b16afd8222b4b349aabd837cb9
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
```

Response

After receiving the request, COS returns the following response, indicating that the inventory tasks list1 and list2 exist in the bucket.
Inventory task list1

- The inventory task analyzes objects with the prefix myPrefix in the bucket examplebucket-1250000000 and all their versions.
- The analysis frequency is once per day.

- Analysis dimensions include Size, LastModifiedDate, StorageClass, ETag, IsMultipartUploaded, and ReplicationStatus.
- The analysis results are stored as a CSV file in the bucket inventorybucket-1250000000. The file has the prefix list1 and is encrypted by using SSE-COS.
  Inventory task list2
- The inventory task analyzes objects with the prefix myPrefix2, and object tag {age:18} in the bucket examplebucket-1250000000 and all their versions.
- The analysis frequency is once per day. Analysis dimensions include Size, LastModifiedDate, StorageClass, ETag, and Tag.
- The analysis results are stored as a CSV file in the bucket inventorybucket-1250000000.
  If there are 100 inventory tasks on a page and IsTruncated is true, COS returns NextContinuationToken, and the value of which can be used as the continuation-token parameter in the GET request to obtain information on the next page.

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 331
Date: Mon, 28 Aug 2018 02:53:39 GMT
x-cos-request-id: NTlhMzg1ZWVfMjQ4OGY3MGFfMWE1NF8****
<?xml version = "1.0" encoding = "UTF-8">
<ListInventoryConfigurationResult xmlns = "http://....">
   <InventoryConfiguration>
      <Id>list1</Id>
      <IsEnabled>True</IsEnabled>
      <Destination>
         <COSBucketDestination>
            <Format>CSV</Format>
            <AccountId>1250000000</AccountId>
            <Bucket>qcs::cos:city::inventorybucket-1250000000</Bucket>
            <Prefix>list1</Prefix>
            <SSE-COS></SSE-COS>
         </COSBucketDestination>
      </Destination>
      <Schedule>
         <Frequency>Daily</Frequency>
      </Schedule>
      <Filter>
         <Prefix>myPrefix</Prefix>
      </Filter>
      <IncludedObjectVersions>All</IncludedObjectVersions>
      <OptionalFields>
         <Field>Size</Field>
         <Field>LastModifiedDate</Field>
```

```
            <Field>ETag</Field>
            <Field>StorageClass</Field>
            <Field>IsMultipartUpload</Field>
            <Field>ReplicationStatus</Field>
        </OptionalFields>
    </InventoryConfiguration>
    <InventoryConfiguration>
        <Id>list2</Id>
        <IsEnabled>True</IsEnabled>
        <Destination>
            <COSBucketDestination>
                <Format>CSV</Format>
                <AccountId>1250000000</AccountId>
                <Bucket>qcs::cos:city::inventorybucket-1250000000</Bucket>
            </COSBucketDestination>
        </Destination>
        <Schedule>
            <Frequency>Weekly</Frequency>
        </Schedule>
        <Filter>
            <And>
                <Prefix>myPrefix2</Prefix>
                <Tag>
                    <Key>age</Key>
                    <Value>18</Value>
                </Tag>
            </And>
        </Filter>
        <IncludedObjectVersions>All</IncludedObjectVersions>
        <OptionalFields>
            <Field>Size</Field>
            <Field>LastModifiedDate</Field>
            <Field>StorageClass</Field>
            <Field>ETag</Field>
            <Field>Tag</Field>
        </OptionalFields>
    </InventoryConfiguration>
    <IsTruncated>false</IsTruncated>
    ------If ContinuationToken was provided in the request---
    <ContinuationToken>...</ContinuationToken>
    <IsTruncated>true</IsTruncated>
    <NextContinuationToken>1ueSDFASDF1Tr/XDAFdadEADadf2J/wm36Hy4vbOwM=</NextContinuati
onToken>
</ListInventoryConfigurationResult>
```

Deleting an Inventory Task

Description

The DELETE Bucket inventory API is used to delete a specified inventory task in a bucket. You should provide the name of the inventory task to be deleted.
For detailed inventory features, see Inventory Feature Overview.

> Note
>
> - To call this API, ensure that you have the necessary permission for inventory tasks in the bucket.
> - The bucket owner has this permission by default. If you do not have it, you should apply for it from the bucket owner first.

Authorization Description

In the authorization policy, action is set to cos:DeleteBucketInventory.

Request

Request Example

```
DELETE /?inventory&id=inventory-configuration-id HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Description
>
> Authorization: Auth String (For details, refer to Request Signature).

Request Parameters

To call the DELETE Bucket inventory API, the inventory task name is required. The format of this parameter is as follows:

Parameter	Description	Type	Required
id	Inventory task name. Default value: None Valid characters: a–z, A–Z, 0–9, -, _, .	String	Yes

Request Header

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

The request body of this request is empty.

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

The response body is empty.

Error Code

This API follows unified error response and error codes. For details, please refer to the Error Codes document.

Practical Case

Request

The following example deletes the list1 task list of the examplebucket-1250000000 .

```
DELETE /?inventory&id=list1 HTTP/1.1
Date: Mon, 28 Aug 2018 02:53:38 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDZfbOAo7cllgPvF9cXFrJD0a1ICvR****&q-sign-time=
1503901499;1503901859&q-key-time=1503901499;1503901859&q-header-list=host&q-url-param-list
=inventory&q-signature=761f3f6449c6a11684464f4b09c6f292f0a4****
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
```

Response

After receiving the request, COS returns a 204 No Content response, indicating that the inventory task list1 in the bucket is deleted.

```
HTTP/1.1 204 No Content
Date: Mon, 28 Aug 2018 02:53:40 GMT
x-cos-id-2:0dfafa/DAPDIFdafdsfDdfSFFfdfKKJdafasiuKJK2
x-cos-request-id: NTlhM2I3M2JfMjQ4OGY3MGFfMWE1NF84****
```

Versioning
Setting Versioning

Description

The PUT Bucket versioning API enables or pauses versioning for a bucket.

Notes

1. If you have never enabled versioning on the bucket, the GET Bucket versioning request does not return a version status value.
2. After enabling versioning, it can only be suspended, not closed.
3. Setting the versioning status value to Enabled or Suspended indicates enabling or suspending versioning.
4. To set the versioning feature for a bucket, you need write permission for the bucket.

Request

Request Example

```
PUT /?versioning HTTP 1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (For details, see Request Signature documentation).

Request Header

Common Headers

This request uses common request headers. For details, see Public Request Headers.

Non-common Headers

This request has no special request header information.

Request Body

```
<VersioningConfiguration>
```

```
  <Status></Status>
</VersioningConfiguration>
```

The data are as follows:

Node Name (Keyword)	Parent Node	Description	Type
VersioningConfiguration	No	Specify the details of versioning.	Container
Status	VersioningConfiguration	Specify whether the versioning is enabled, enumerated values: Suspended, Enabled.	Enum

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Common Response Headers.

Special Response Headers

There are no special response headers for this response operation.

Response Body

The response body is empty.

Error Analysis

The following table describes some special and common errors that may occur with this request. For details on error codes related to COS or the complete list of errors for the product, see Error Codes.

Error Code	HTTP Status Code	Description
InvalidArgument	400 Bad Request	If the XML body is empty when versioning is enabled, it will return InvalidArgument.
InvalidDigest	400 Bad Request	1. The Content-MD5 carried is inconsistent with the request body calculated by the server. 1. There are only two valid values for the state of versioning: Enabled and Suspended. If any other state is specified, it will return InvalidArgument.

Practical Case

Request

```
PUT /?versioning HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: python-requests/2.12.4
Content-Type: application/xml
Authorization: q-sign-algorithm=sha1&q-ak=AKID15IsskiBQKTZbAo6WhgcBqVls9Sm****&q-sign-tim
e=1480932292;1981012292&q-key-time=1480932292;1981012292&q-url-param-list=versioning&q-h
eader-list=host&q-signature=47ec2b80c73788ecd394d3b9ad90e120a32f****
Content-Length: 83

<VersioningConfiguration>
   <Status>Enabled</Status>
</VersioningConfiguration>
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Connection: keep-alive
Date: Wed, 23 Aug 2017 08:14:53 GMT
x-cos-request-id: NTk5ZDM5N2RfMjNiMjM1MGFfMmRiX2Y0****
```

Querying Versioning Information

Description

The GET Bucket versioning API is used to obtain versioning information for a bucket.

Notes

1. To obtain the status of versioning for a bucket, read permission for the bucket is required.

2. There are three versioning states: versioning not enabled, versioning enabled, and versioning suspended.

    - If you have never enabled (or suspended) versioning on the bucket, the response is:

```
<VersioningConfiguration/>
```

    - If you have enabled versioning for the bucket, the response is:

```
<VersioningConfiguration>
  <Status>Enabled</Status>
</VersioningConfiguration>
```

    - If you have suspended versioning for the bucket, the response is:

```
<VersioningConfiguration>
  <Status>Suspended</Status>
</VersioningConfiguration>
```

Request

Request Example

```
GET /?versioning HTTP 1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (For details, see Request Signature documentation).

Request Header

Common Headers

This request uses common request headers. For details, see Public Request Headers.

Non-common Headers

This request has no special request header information.

Request Body

The request does not have a request body.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Common Response Headers.

Special Response Headers

There are no special response headers for this response operation.

Response Body

```
<VersioningConfiguration>
   <Status></Status>
</VersioningConfiguration>
```

The data are as follows:

Node Name (Keyword)	Parent Node	Description	Type
VersioningConfiguration	No	Specify the details of versioning.	Container
Status	VersioningConfiguration	Specify whether the versioning is enabled, enumerated values: Suspended, Enabled.	Enum

Practical Case

Request

```
GET /?versioning HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: python-requests/2.12.4
Authorization: q-sign-algorithm=sha1&q-ak=AKID15IsskiBQKTZbAo6WhgcBqVls9Sm****&q-sign-tim
e=1480932292;1981012292&q-key-time=1480932292;1981012292&q-url-param-list=versioning&q-h
eader-list=host&q-signature=5118a936049f9d44482bbb61309235cf4abe****
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 120
Connection: keep-alive
Date: Wed, 23 Aug 2017 08:15:16 GMT
x-cos-request-id: NTk5ZDM5OTRfZDNhZDM1MGFfMjYyMTFfZmU3****

<?xml version='1.0' encoding='utf-8' ?>
<VersioningConfiguration>
    <Status>Enabled</Status>
</VersioningConfiguration>
```

In-cloud Replication (replication) Setting Cross-Region Replication

Description

The PUT Bucket replication API is used to configure cross-region replication rules for a bucket with versioning enabled. If cross-region replication rules are already configured for the bucket, this request will replace the existing settings.

> Note:
>
> To call this API, ensure that versioning is enabled for the bucket. For details about the API for enabling versioning, see PUT Bucket versioning.

Request

Request Example

```
PUT /?replication HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Content-MD5: MD5
Authorization: Auth String
request body
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Header

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

Set specific cross-region replication configuration in the request body. The configuration includes the enabling status of the cross-region replication rule, content to be copied, and the name, storage region, and other information of the destination bucket. For each bucket with versioning enabled, COS supports only one cross-region replication rule.

```
<ReplicationConfiguration>
  <Role>qcs::cam::uin/<OwnerUin>:uin/<SubUin></Role>
  <Rule>
    <Status></Status>
    <ID></ID>
    <Prefix></Prefix>
    <Destination>
      <Bucket>qcs::cos:<Region>::<BucketName-APPID></Bucket>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

The specific contents are described as follows:

Node Name (Keyword)	Parent Node	Description	Type	Required
ReplicationConfiguration	No	All cross-region replication configurations.	Container	Yes

Node Name (Keyword)	Parent Node	Description	Type	Required
Role	ReplicationConfiguration	Identifier of the initiator: qcs::cam::uin/<OwnerUin>:uin/<SubUin>	String	Yes
Rule	ReplicationConfiguration	Specific configuration information. Up to 1000 rules are supported.	Container	Yes
ID	ReplicationConfiguration.Rule	Identifies the name of a specific rule.	String	No
Status	ReplicationConfiguration.Rule	Identifies whether a rule takes effect. Enumerated values: Enabled and Disabled	String	Yes
Prefix	ReplicationConfiguration.Rule	Prefix matching policy, which cannot overlap. If overlapping occurs, an error is returned. The root directory for prefix matching is empty.	String	Yes
Destination	ReplicationConfiguration.Rule	Destination bucket information.	Container	Yes
Bucket	ReplicationConfiguration.Rule.Destination	Resource identifier: qcs::cos:<Region>::<BucketName-APPID>	String	Yes
StorageClass	ReplicationConfiguration.Rule.Destination	Storage class. Enumerated values: STANDARD and STANDARD_IA; default value: original bucket class Note: Cross-region replication does not support specifying copied objects as the archive storage. To set the object replicas as the archive storage type, configure lifecycle management in the destination bucket. For detailed operations, see PUT Bucket lifecycle.	String	No

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

The response body is empty.

Error Analysis

The following table describes some common special errors that may occur with this request. For all error information, see Error Code.

Error Code	Description	Status Code
InvalidBucketState	Versioning is not enabled for the bucket. As a result, cross-region replication cannot be enabled.	409 Conflict
InvalidArgument	The parameter content is invalid.	400 Bad Request

Practical Case

Request

The following PUT Bucket replication request adds a cross-region replication configuration record to the bucket originbucket-1250000000 . The configuration is used to copy objects with the prefix testPrefix to the destination bucket destinationbucket-1250000000 in Region1.

```
PUT /?replication HTTP/1.1
Date: Mon, 28 Aug 2017 02:53:38 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDZfbOAo7cllgPvF9cXFrJD0a1ICvR**&q-sign-time=1503888878;1503889238&q-key-time=1503888878;1503889238&q-header-list=host&q-url-param-list=replication&q-signature=254bf9cd3d6615e89a36ab652437f9d45c5f**
Content-MD5: AAq9nzrpsz5LJ4UEe1f6Q==
Host: <BucketName-APPID>.<Endpoint>
Content-Length: 312

<ReplicationConfiguration>
  <Role>qcs::cam::uin/100000000001:uin/100000000001</Role>
  <Rule>
    <Status>Enabled</Status>
    <ID>RuleId_01</ID>
    <Prefix>testPrefix</Prefix>
    <Destination>
      <Bucket>qcs::cos:ap-region1::destinationbucket-1250000000</Bucket>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

Response

After receiving the request, COS returns the following response, indicating that the cross-region replication configuration is set successfully.

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Connection: keep-alive
Date: Fri, 14 Apr 2019 07:06:19 GMT
x-cos-bucket-region: ap-region1
x-cos-request-id: NWQwMzQ3NmJfMjRiMjU4NjRfOTM4NV82ZDU1****
x-cos-trace-id: OGVmYzZiMmQzYjA2OWNhODk0NTRkMTBiOWVmMDAxODc0OWRkZjk0ZDM1NmI1M2E2MTRlY2MzZDhmNmI5MWI1OWE4OGMxZjNjY2JiNTBmMTVmMWY1MzAzYzkyZGQ2ZWM4MzUyZTg1NGRhNWY0NTJiOGUyNTViYzgyNzgxZTEwOTY=
```

Querying Cross-Region Replication

Description

The GET Bucket replication API is used to query the cross-region replication configuration in a bucket. When initiating the request, you need to obtain a request signature, indicating that the request is authorized.

Request

Request Example

```
GET /?replication HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (For details, see Request Signature documentation).

Request Header

Common Headers

This request uses common request headers. For details, see Public Request Headers.

Non-common Headers

This request has no special request header information.

Request Body

The request does not have a request body.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Common Response Headers.

Special Response Headers

Parameter	Type	Description
x-cos-replication-rule-creation-time	UTC timestamp format.	Creation time of a cross-region replication rule.

Response Body

The response body is returned as application/xml. An example including the complete node data is shown below:

```
<ReplicationConfiguration>
  <Role>qcs::cam::uin/[UIN]:uin/[Subaccount]</Role>
  <Rule>
    <Status></Status>
    <ID></ID>
    <Prefix></Prefix>
    <Destination>
      <Bucket>qcs::cos:[Region]::[BucketName-APPID]</Bucket>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

The specific contents are described as follows:

Node Name (Keyword)	Parent Node	Description	Type	Required
ReplicationConfiguration	No	All cross-region replication configurations.	Container	Yes
Role	ReplicationConfiguration	Identifier of the initiator. qcs::cam::uin/<OwnerUin>:uin/<SubUin>	String	Yes
Rule	ReplicationConfiguration	Specific configuration. You can set a maximum of 1,000 rules, which should apply to the same destination bucket.	Container	Yes
ID	ReplicationConfiguration.Rule	Identifies the name of a specific rule.	String	No
Status	ReplicationConfiguration.Rule	Indicates whether a rule takes effect. Enumerated values: Enabled and Disabled	String	Yes
Prefix	ReplicationConfiguration.Rule	Prefix matching policy. Policies cannot overlap; otherwise, an error will be returned. To match the root	String	Yes

Node Name (Keyword)	Parent Node	Description	Type	Required
		directory, leave this parameter empty.		
Destination	ReplicationConfiguration.Rule	Destination bucket information.	Container	Yes
Bucket	ReplicationConfiguration.Rule.Destination	Resource identifier: qcs::cos:[region]::[bucketname-Appid]	String	Yes
StorageClass	ReplicationConfiguration.Rule.Destination	Storage class. Enumerated values: Standard and Standard_IA; default value: original bucket class	String	No

Error Analysis

The following table describes some common special errors that may occur with this request. For common error information, see Error Code.

Error Code	Description	Status Code
ReplicationConfigurationNotFoundError	The configuration for cross-region replication is not found.	404 Not Found

Practical Case

Request

The following example queries the cross-region replication configuration from the originbucket-1250000000 .

```
GET /?replication HTTP/1.1
Date: Fri, 14 Apr 2019 07:17:19 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=1503895278;1503895638&q-key-time=1503895278;1503895638&q-header-list=host&q-url-param-list=replication&q-signature=f77900be432072b16afd8222b4b349aabd837cb9
Host: <BucketName-APPID>.<Endpoint>
Content-Length: 0
```

Response

After receiving the request, COS returns the following response, indicating that the cross-region replication configuration in the bucket is enabled. In the rule configuration, the copied content is all objects with the prefix testPrefix in the bucket

originbucket-1250000000 . By default, the storage type of object replicas is the same as that of the objects in the source bucket.

```
Content-Type: application/xml
Content-Length: 309
Connection: keep-alive
Date: Fri, 14 Apr 2019 07:17:19 GMT
x-cos-replication-rule-creation-time: Fri, 14 Apr 2019 07:06:19 GMT
x-cos-request-id: NWQwMzQ5ZmZfMjBiNDU4NjRfNjAwOV84MzA2MjE=
<ReplicationConfiguration>
    <Role>qcs::cam::uin/100000000001:uin/100000000001</Role>
    <Rule>
        <Status>Enabled</Status>
        <ID>RuleId_01</ID>
        <Prefix>testPrefix</Prefix>
        <Destination>
            <Bucket>qcs::cos:ap-[Region]::destinationBucket-1250000000</Bucket>
        </Destination>
    </Rule>
</ReplicationConfiguration>
```

Deleting In-cloud Replication Rules

Description

The DELETE Bucket replication API is used to delete the in-cloud replication configuration in a bucket. When initiating the request, users should obtain a request signature, indicating that the request is authorized.

Request

Request Example

```
DELETE /?replication HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (For details, see Request Signature documentation).

Request Header

Common Headers

This request uses common request headers. For details, see Public Request Headers.

Non-common Headers

This request has no special request header information.

Request Body

The request body of this request is empty.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Common Response Headers.

Special Response Headers

There are no special response headers for this response operation.

Response Body

The response body is empty.

Practical Case

Request

The following request example shows the deletion of cross-region configuration information from the bucket originBucket-1250000000 .

```
DELETE /?replication HTTP/1.1
Date: Fri, 14 Apr 2019 07:47:35 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=15
03901499;1503901859&q-key-time=1503901499;1503901859&q-header-list=host&q-url-param-list=r
eplication&q-signature=761f3f6449c6a11684464f4b09c6f292f0a4e7e0
Host: originBucket-1250000000.cos.city.yfm4.fsphere.cn
```

Response

After receiving the request, COS returns a 204 No Content response, indicating that the in-cloud replication configuration in the bucket is deleted. After the in-cloud replication configuration is deleted, COS no longer copies objects from the source bucket to the destination bucket. However, the object data already in the destination bucket is retained.

```
Content-Length: 0
Connection: keep-alive
Date: Fri, 14 Apr 2019 07:47:35 GMT
x-cos-request-id: NWQwMzUxMTdfMjBiNDU4NjRfNWZlZF84MjdmZTE=
x-cos-trace-id: OGVmYzZiMmQzYjA2OWNhODk0NTRkMTBiOWVmMDAxODc0OWRkZjk0ZDM1NmI1
M2E2MTRlY2MzZDhmNmI5MWI1OWE4OGMxZjNjY2JiNTBmMTVmMWY1MzAzYzkyZGQ2ZWM4MzUy
ZTg1NGRhNWY0NTJiOGUyNTViYzgyNzgxZTEwOTY=
```

Cross-Cloud Replication (CrossCloudReplication) Setting Cross-Cloud Replication Rules

Feature Description

The PUT Bucket CrossCloudReplication API is used to create, update, and delete cross-cloud replication rules for the source bucket. If the bucket has been configured with cross-cloud replication rules, this request will update the corresponding rules.

Request

First, obtain the cross-cloud replication request domain name. In a private cloud environment, log in to the platform machine and run the following command to obtain it:

```
[                    ~]$ kubectl get ingress -A | grep cos-laz
tce                ocloud-cos-lazarus                                    nginx    ocloud-cos-lazarus.                    .fsphere.cn
                                                                                                                  80        145d
```

Request Example

```
POST /capi HTTP/1.1
Host: ocloud-cos-lazarus.xxxx.fsphere.cn
Accept: */*
Content-Type: application/json
request body
```

Request Body

Users can set specific configuration information for cross-cloud replication in the request body. The configuration information includes the enabling status of cross-cloud replication rules, replication content, the bucket name of the destination bucket, AK/SK, and other information.

```
{
  "Action": "PutBucketCrossCloudReplication",
  "Params": {
    "Bucket": "test-1-1255000000",
    "Region": "region",
    "Configuration": [
```

```
      {
        "Destination": {
          "Bucket": "test-2-1255000000",
          "Endpoint": "http://cos.region.domain",
          "SecretID": "AKIDQDXxxxxxxxxxxxxxxxxxxqc5ltHpjJf",
          "SecretKey": "nyxljpwoxxxxxxxxxxxxxxxxxxTRwkgLU"
        },
        "DestinationPrefix": "test",
        "SourcePrefix": "test/",
        "Status": "Enabled",
        "PrefixType": true
      }
    ]
  }
}
```

The specific content is described as follows:

Field Name	Description	Type	Required
Action	Operation type for cross-cloud replication rules.	String	Yes
Params	Configuration information on cross-cloud replication rules.	Dict	Yes

Field information contained in Params.

Field Name	Description	Type	Required
Bucket	Source bucket name.	String	Yes
Region	Source bucket region information.	String	Yes
Configuration	Specific rule configuration information.	Array	Yes

Field information contained in Configuration.

Field Name	Description	Type	Required
Destination	Destination bucket configuration information.	Dict	Yes
DestinationPrefix	Prefix information configured for the destination bucket.	String	Yes
SourcePrefix	Prefix information configured for the source bucket.	Dict	Yes
PrefixType	Allowlist/Blocklist type configured for the source bucket. true indicates that the blocklist is enabled (which means not copying objects with the prefix specified by	bool	Yes

Field Name	Description	Type	Required
	SourcePrefix), and false indicates that the allowlist is enabled (which means copying only objects with the prefix specified by SourcePrefix).		
Status	Whether the cross-cloud replication rule is enabled.	String	Yes

Field information contained in Destination.

Field Name	Description	Type	Required
Bucket	Destination bucket name.	String	Yes
Endpoint	Destination bucket access domain name, excluding the bucket name.	String	Yes
SecretID	Destination bucket AK information.	String	Yes
SecretKey	Destination bucket SK information.	String	Yes

Response

Response Header

```
HTTP/1.1 200 OK
Date: Tue, 23 Sep 2025 07:58:40 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 810
Connection: keep-alive
```

Response Body

```
{
  "RequestId": "",
  "Code": "ok",
  "Message": "",
  "Data": [
    {
      "Status": "Enabled",
      "Destination": {
        "Endpoint": "http://cos.region.domain",
        "Bucket": "test-2-1255000000",
        "SecretID": "AKIDQDXnxxxxxxxxxxxxxxxqc5ltHpjJf",
```

```
        "SecretKey": "nyxljpwolxxxxxxxxxxxx50pTRwkgLU",
        "SecretToken": ""
      },
      "SourcePrefix": "test/",
      "DestinationPrefix": "test/",
      "Concurrency": 0,
      "RequestLimit": 0,
      "TrafficLimit": 0,
      "PrefixType": false
    }
  ]
}
```

Querying Cross-Cloud Replication Rules

Feature Description

The GET Bucket CrossCloudReplication API is used to query the cross-cloud replication rules configured for the source bucket.

Request

First, obtain the cross-cloud replication request domain name. In a private cloud environment, log in to the platform machine and run the following command to obtain it:

```
[                    ~]$ kubectl get ingress -A | grep cos-laz
tce                ocloud-cos-lazarus                                    nginx    ocloud-cos-lazarus.                    .fsphere.cn
                                                                                                                  80        145d
```

Request Example

```
POST /capi HTTP/1.1
Host: ocloud-cos-lazarus.xxxx.fsphere.cn
Accept: */*
Content-Type: application/json
request body
```

Request Body

Users can specify information for querying the bucket in the request body.

```
{
  "Action": "GetBucketCrossCloudReplication",
  "Appid": 1255000000,
  "Params": {
    "Bucket": "test-1-1255000000",
    "Region": "region"
  }
}
```

Response

Response Header

```
HTTP/1.1 200 OK
Date: Tue, 23 Sep 2025 07:58:40 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 810
Connection: keep-alive
```

Response Body


```
{
  "RequestId": "",
  "Code": "ok",
  "Message": "",
  "Data": [
    {
      "Status": "Enabled",
      "Destination": {
        "Endpoint": "http://cos.region.domain",
        "Bucket": "test-2-1255000000",
        "SecretID": "AKIDQDXnxxxxxxxxxxxxxxxqc5ltHpjJf",
        "SecretKey": "nyxljpwolxxxxxxxxxxxxx50pTRwkgLU",
        "SecretToken": ""
      },
      "SourcePrefix": "test/",
      "DestinationPrefix": "test/",
      "Concurrency": 0,
      "RequestLimit": 0,
      "TrafficLimit": 0,
      "PrefixType": false
    }
  ]
}
```

Deleting Cross-Cloud Replication Rules

Feature Description

The PUT Bucket CrossCloudReplication API is used to create, update, and delete cross-cloud replication rules configured for the source bucket.

Request

First, obtain the cross-cloud replication request domain name. In a private cloud environment, log in to the platform machine and run the following command to obtain it:

```
[ ~]$ kubectl get ingress -A | grep cos-laz
tce ocloud-cos-lazarus nginx ocloud-cos-lazarus. .fsphere.cn
                                                  80 145d
```

Request Example

```
POST /capi HTTP/1.1
Host: ocloud-cos-lazarus.xxxx.fsphere.cn
Accept: */*
Content-Type: application/json
request body
```

Request Body

Users can delete a rule by setting the specific configuration information for cross-cloud replication to empty in the request body. If there are multiple cross-cloud replication rules, users only need to fill in the Configuration field with the other rules that need to be retained to delete the specified rule. In other words, the creation and deletion operations share the same API, and the existing rules in Configuration represent the rules that will be created or modified.

```
{
  "Action": "PutBucketCrossCloudReplication",
  "Params": {
    "Bucket": "test-1-1255000000",
    "Region": "region",
    "Configuration": []
  }
}
```

Response

Response Header

```
HTTP/1.1 200 OK
Date: Tue, 23 Sep 2025 07:58:40 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 810
Connection: keep-alive
```

Response Body

```
{
  "RequestId": "",
  "Code": "ok",
  "Message": "",
  "Data": []
}
```

Logging
Setting Log Management

Description

The PUT Bucket Logging API is used to enable logging for the source bucket to save the access logs of the source bucket into a specified destination bucket.

> Note:
>
> - Only the owner of the source bucket can call this API.
> - To enable the access logging feature, you need to grant CLS the permission to write data into COS. For the authorization process, see Enable log management.

Request

Request Example

```
PUT /?logging HTTP 1.1
Host: <BucketName-APPID>.<Endpoint>
Date:date
Content-Length: length
Content-Type: application/xml
Content-MD5: MD5
Authorization: Auth String
```

> Note:
>
> - <BucketName-APPID> is the bucket name with APPID as the suffix, for example, examplebucket-1250000000.
> - Authorization: Auth String (For details, see Request Signature documentation).

Request Header

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

A request body is required to implement this request operation. The content of an example request body with all nodes is as follows:

```
<BucketLoggingStatus>
```

```
  <LoggingEnabled>
    <TargetBucket>examplebucket-1250000000</TargetBucket>
    <TargetPrefix>prefix</TargetPrefix>
  </LoggingEnabled>
</BucketLoggingStatus>
```

The data are described as follows:

Node Name (Keyword)	Parent Node	Description	Type	Required
BucketLoggingStatus	No	Logging status. If there is no child node information, logging is disabled.	Container	Yes

Content of the BucketLoggingStatus node of the Container type:

Node Name (Keyword)	Parent Node	Description	Type	Required
LoggingEnabled	BucketLoggingStatus	Specific logging settings information of a bucket, mainly the destination bucket.	Container	No

Content of the LoggingEnabled node of the Container type:

Node Name (Keyword)	Parent Node	Description	Type	Required
TargetBucket	LoggingEnabled	Destination bucket for storing logs, which can be the same bucket (not recommended), or a bucket under the same account in the same region.	String	No
TargetPrefix	LoggingEnabled	Path in the destination bucket that stores logs.	String	No

> Note:
>
> After you specify the bucket and path for storing logs, the generated log file name is in Destination bucket/path prefix{YYYY}/{MM}/{DD}/{time}_{random}_{index}.gz format.

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

The response body is empty.

Error Code

This API follows unified error response and error codes. For details, please refer to the Error Codes document.

Practical Case

Request

```
PUT /?logging HTTP 1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Fri, 10 Mar 2017 09:45:46 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDWtTCBYjM5OwLB9CAwA1Qb2ThTSUj****&q-sign-time=1484814927;32557710927&q-key-time=1484814927;32557710927&q-header-list=host&q-url-param-list=accelerate&q-signature=8b9f05dabce2578f3a79d732386e7cbade90****
Content-Type: application/xml
Content-Length: 147

<BucketLoggingStatus>
  <LoggingEnabled>
    <TargetBucket>examplebucket-1250000000</TargetBucket>
    <TargetPrefix>prefix</TargetPrefix>
  </LoggingEnabled>
</BucketLoggingStatus>
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Connection: keep-alive
Date: Fri, 10 Mar 2017 09:45:46 GMT
x-cos-request-id: NTg4MDdiZWRfOWExZjRlXzQ2OWVf****
```

Querying Log Management

Description

The GET Bucket logging API is used to obtain the log configuration of the source bucket.

> Note:
>
> Only the owner of the source bucket can call this API.

Request

Request Example

```
GET /?logging HTTP 1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> - <BucketName-APPID> is the bucket name with APPID as the suffix, for example, examplebucket-1250000000.
> - Authorization: Auth String (For details, see Request Signature documentation).

Request Header

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

The request body of this request is empty.

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

The response body returns application/xml data. The following contains all the nodes:

```
<BucketLoggingStatus>
  <LoggingEnabled>
    <TargetBucket>examplebucket-1250000000</TargetBucket>
    <TargetPrefix>prefix</TargetPrefix>
  </LoggingEnabled>
</BucketLoggingStatus>
```

The data are as follows:

Node Name (Keyword)	Parent Node	Description	Type
BucketLoggingStatus	No	Log status of the bucket.	Container

Content of the BucketLoggingStatus node of the Container type:

Node Name (Keyword)	Parent Node	Description	Type
LoggingEnabled	BucketLoggingStatus	Detailed logging configuration of the bucket.	Container

Content of the LoggingEnabled node of the Container type:

Node Name (Keyword)	Parent Node	Description	Type
TargetBucket	LoggingEnabled	Destination bucket for storing logs, which can be the same bucket (not recommended), or a bucket under the same account in the same region.	String
TargetPrefix	LoggingEnabled	Path in the destination bucket that stores logs.	String

Error Code

This API follows unified error response and error codes. For details, please refer to the Error Codes document.

Practical Case

Request

```
GET /?logging HTTP 1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Wed, 28 Oct 2017 21:32:00 GMT
```

```
Authorization: q-sign-algorithm=sha1&q-ak=AKIDWtTCBYjM5O****&q-sign-time=1484815944;32557
711944&q-key-time=1484815944;32557711944&q-header-list=host&q-url-param-list=accelerate&q-s
ignature=a2d28e1b9023d09f9277982775a4b3b705d0****
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 142
Connection: keep-alive
Date: Wed, 28 Oct 2017 21:32:00 GMT
x-cos-request-id: NTg4MDdlNGZfNDYyMDRlXzM0YWFf****

<BucketLoggingStatus>
  <LoggingEnabled>
    <TargetBucket>examplebucket-1250000000</TargetBucket>
    <TargetPrefix>prefix</TargetPrefix>
  </LoggingEnabled>
</BucketLoggingStatus>
```

Bucket Encryption
Setting Bucket Encryption

Description

The PUT Bucket encryption API is used to set the default encryption configuration for a specified bucket.

To execute this API, PutBucketEncryption permission is required. By default, the owner of the bucket has permission to use this API directly, and the bucket owner can also grant permission to other users.

Request

Request Sample

```
PUT /?encryption HTTP 1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> - <BucketName-APPID> is the bucket name with APPID as the suffix, for example, examplebucket-1250000000.
> - Authorization: Auth String (For details, refer to Request Signature).

Request Parameters

This API has no request parameters.

Request Headers

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

Users use XML language in the request body to set bucket default encryption configuration information. Encryption configuration information mainly consists of encryption items.

The following is the request body used to set SSE-COS:

```
<ServerSideEncryptionConfiguration>
    <Rule>
      <ApplyServerSideEncryptionByDefault>
        <SSEAlgorithm>AES256|KMS</SSEAlgorithm>
        <KMSMasterKeyID>String</KMSMasterKeyID>
      </ApplyServerSideEncryptionByDefault>
    </Rule>
</ServerSideEncryptionConfiguration>
```

Specific elements are as follows:

Element Name	Parent Node	Description	Type	Required

Element Name	Parent Node	Description	Type	Required
ServerSideEncryptionConfiguration	No	Configuration parameters including default encryption	Container	Yes
Rule	ServerSideEncryptionConfiguration	Default server-side encryption configuration rules	Container	Yes
ApplyServerSideEncryptionByDefault	ServerSideEncryptionConfiguration.Rule	Default configuration information for server-side encryption	Container	Yes
SSEAlgorithm	ServerSideEncryptionConfiguration.Rule. ApplyServerSideEncryptionByDefault	Supported enumeration values are AES256 and KMS. AES256 indicates that the SSE-COS mode is used and the encryption algorithm is AES256. KMS indicates the SSE-KMS mode.	String	Yes
KMSMasterKeyID	ServerSideEncryptionConfiguration.Rule. ApplyServerSideEncryptionByDefault	CMK of the KMS when the value of SSEAlgorithm is KMS. If it is not specified, the CMK created by COS by default is used. For details, see SSE-KMS Encryption.	String	Yes

Response

Response Headers

This API only returns common response headers. For details, please refer to the Public Response Headers document.

Response Body

The response body for this request is empty.

Error Codes

This API follows unified error response and error codes. For details, please refer to the Error Codes document.

Practical Case

Request

The following example demonstrates setting SSE-COS encryption for the bucket examplebucket-1250000000.

```
PUT /?encryption HTTP 1.1
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Date: Mon, 17 Jun 2019 08:37:35 GMT
Authorization: signatureValue

<ServerSideEncryptionConfiguration>
    <Rule>
      <ApplyServerSideEncryptionByDefault>
         <SSEAlgorithm>AES256</SSEAlgorithm>
      </ApplyServerSideEncryptionByDefault>
    </Rule>
</ServerSideEncryptionConfiguration>
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Date: Mon, 17 Jun 2019 08:37:36 GMT
x-cos-request-id: NWQwNzUxNTBfMzdiMDJhMDlfOWM0Nl85NDFk****
```

Querying Bucket Encryption

Description

The GET Bucket encryption API is used to query the default encryption configuration for a specified bucket.

To execute this API, GetBucketEncryption permission is required. By default, the owner of the bucket has permission to use this API directly, and the bucket owner can also grant permission to other users.

Request

Request Example

```
GET /?encryption HTTP 1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> - is the bucket name with APPID as the suffix, for example, examplebucket-1250000000.
> - Authorization: Auth String (For details, see Request Signature).

Request Parameters

This API has no request parameters.

Request Headers

This API only uses common request headers. For details, please refer to the Common Request Headers document.

Request Body

This API has no request body.

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

The SSE-COS-encrypted response body is as follows:

```
<ServerSideEncryptionConfiguration>
    <Rule>
        <ApplyServerSideEncryptionByDefault>
            <SSEAlgorithm>AES256|KMS</SSEAlgorithm>
            <KMSMasterKeyID>String</KMSMasterKeyID>
        </ApplyServerSideEncryptionByDefault>
    </Rule>
</ServerSideEncryptionConfiguration>
```

Specific elements are as follows:

Element Name	Parent Node	Description	Type
ServerSideEncryptionConfiguration	No	Configuration parameters including default encryption	Container

Element Name	Parent Node	Description	Type
Rule	ServerSideEncryptionConfiguration	Default server-side encryption configuration rules	Container
ApplyServerSideEncryptionByDefault	ServerSideEncryptionConfiguration.Rule	Default configuration information for server-side encryption	Container
SSEAlgorithm	ServerSideEncryptionConfiguration.Rule.ApplyServerSideEncryptionByDefault	Supported enumeration values are AES256 and KMS. AES256 indicates that the SSE-COS mode is used and the encryption algorithm is AES256. KMS indicates the SSE-KMS mode.	String
KMSMasterKeyID	ServerSideEncryptionConfiguration.Rule.ApplyServerSideEncryptionByDefault	CMK of the KMS when the value of SSEAlgorithm is KMS. If it is not specified, the CMK created by COS by default is used. For details, see SSE-KMS Encryption.	String

Error Codes

This API follows unified error response and error codes. For details, please refer to the Error Codes document.

Practical Case

Request

```
GET /?encryption HTTP 1.1
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Date: Mon, 17 Jun 2019 08:37:35 GMT
Authorization: signatureValue
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: xxxx
Date: Mon, 17 Jun 2019 08:37:36 GMT
x-cos-request-id: NWQwNzUxNTBfMzdiMDJhMDlfOWM0Nl85NDFk****
```

```
<?xml version = "1.0" encoding = "UTF-8">
<ServerSideEncryptionConfiguration>
    <Rule>
        <ApplyServerSideEncryptionByDefault>
            <SSEAlgorithm>AES256</SSEAlgorithm>
        </ApplyServerSideEncryptionByDefault>
    </Rule>
</ServerSideEncryptionConfiguration>
```

Deleting Bucket Encryption

Description

The DELETE Bucket encryption API is used to delete the default encryption configuration for a specified bucket.

To execute this API, DeleteBucketEncryption permission is required. By default, the owner of the bucket has permission to use this API directly, and the bucket owner can also grant permission to other users.

Request

Request Example

```
DELETE /?encryption HTTP 1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> - <BucketName-APPID> is the bucket name with APPID as the suffix, for example, examplebucket-1250000000.
> - Authorization: Auth String (For details, refer to Request Signature).

Request Parameters

This API has no request parameters.

Request Headers

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

The request body of this request is empty.

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers

documentation.

Response Body

The response body for this request is empty.

Error Codes

This API follows unified error response and error codes. For details, please refer to the Error Codes document.

Practical Case

Request

The following example demonstrates deleting the default SSE-COS encryption configuration from the bucket examplebucket-1250000000.

```
DELETE /?encryption HTTP 1.1
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Date: Mon, 17 Jun 2019 08:37:35 GMT
Authorization: signatureValue
```

Response

```
HTTP/1.1 204 No Content
Date: Mon, 17 Jun 2019 08:37:36 GMT
x-cos-request-id: NWQwNzUxNTBfMzdiMDJhMDlfOWM0Nl85NDFk****
```

Object Locking
Setting Object Lock

Description

COS allows you to set object locking for existing buckets. The PUT Bucket ObjectLockConfiguration API is used to set the object locking feature for a bucket to meet compliance requirements.

Request

Request Example

```
PUT /?object-lock HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> - <BucketName-APPID> is the bucket name with APPID as the suffix, for example, examplebucket-1250000000.
> - Authorization: Auth String (For details, refer to Request Signature).

Request Header

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

```
<?xml version="1.0" encoding="UTF-8" ?>
<ObjectLockConfiguration>
    <ObjectLockEnabled>Enabled</ObjectLockEnabled>
        <Rule>
            <DefaultRetention>
                <Days>30</Days>
            </DefaultRetention>
        </Rule>
    </ObjectLockConfiguration>
```

The detailed data is described as follows:

Node Name (Keyword)	Parent Node	Description	Type	Required
ObjectLockConfiguration	No	Object locking configuration	Container	Yes
ObjectLockEnabled	ObjectLockConfiguration	Whether object locking is enabled	String	Yes

Node Name (Keyword)	Parent Node	Description	Type	Required
Rule	ObjectLockConfiguration	Object locking rule	Containers	Yes
DefaultRetention	ObjectLockConfiguration.Rule	Default object locking period	Containers	Yes
Days	ObjectLockConfiguration.Rule.DefaultRetention	Duration of the default object locking period (1–36500)	Int	Yes

> Note:
>
> - The ObjectLockEnabled parameter can only be set to Enabled. When it is set to Enabled, the object locking feature is enabled for the bucket.
> - Once the object locking feature is enabled, it cannot be disabled. Please proceed with caution.

Response

Response Headers

This API only returns common response headers. For details, please refer to the Public Response Headers document.

Response Body

The response body is empty.

Error Code

This API follows uniform error responses and error codes. For error codes other than the following error information, see Error Code.

HTTP Status Code	Error Code	Description
409 Conflict	InvalidLockedTime	If the value of Days is less than the original time, the following error is returned: The object locking time of the bucket cannot be less than the original time, and the value should be 1–36500 days.

Practical Case

Request

The following example sets object locking for the bucket examplebucket-1250000000, with a retention period of 1 day.

```
PUT /?object-lock= HTTP/1.1
Host: exmaplebucket-1250000000.cos.city.yfm4.fsphere.cn
```

```
Content-Length: 281
Content-Type: application/x-www-form-urlencoded
Authorization: Auth String

<ObjectLockConfiguration>
  <ObjectLockEnabled>Enabled</ObjectLockEnabled>
  <Rule>
    <DefaultRetention>
      <Days>1</Days>
    </DefaultRetention>
  </Rule>
</ObjectLockConfiguration>
```

Response

```
HTTP/1.1 200 OK
Content-Length: 0
Connection: keep-alive
Date: Fri, 09 Dec 2022 08:17:20 GMT
x-cos-request-id: NjM5MmVmMTBfNmM0ZTQ0MGJfMjA4****
```

Querying the Object Lock

Description

COS allows you set object locking for existing buckets. The GET Bucket ObjectLockConfiguration API is used to obtain the effective object locking configuration.

Request

Request Example

```
GET /?object-lock HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> - <BucketName-APPID> is the bucket name with APPID as the suffix, for example, examplebucket-1250000000.
> - Authorization: Auth String (For details, refer to Request Signature).

Request Header

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

The request body of this request is empty.

Response

Response Headers

This API only returns common response headers. For details, please refer to the Public Response Headers document.

Response Body

```
<?xml version="1.0" encoding="UTF-8" ?>
<ObjectLockConfiguration>
    <ObjectLockEnabled>Enabled</ObjectLockEnabled>
    <Rule>
```

```
        <DefaultRetention>
            <Days>30</Days>
        </DefaultRetention>
    </Rule>
</ObjectLockConfiguration>
```

The detailed data is described as follows:

Node Name (Keyword)	Parent Node	Description	Type
ObjectLockConfiguration	No	Object locking configuration	Container
ObjectLockEnabled	ObjectLockConfiguration	Whether object locking is enabled	String
Rule	ObjectLockConfiguration	Object locking rule	Containers
DefaultRetention	ObjectLockConfiguration.Rule	Default object locking period	Containers
Days	ObjectLockConfiguration.Rule.DefaultRetention	Duration of the default object locking period (1–36500 days)	Int

Error Code

This API follows unified error response and error codes. For details, please refer to the Error Codes document.

Practical Case

Request

The following example obtains the object locking configuration of the bucket examplebucket-1250000000.

```
GET /?object-lock= HTTP/1.1
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Authorization: Auth String
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 180
Connection: keep-alive
Date: Fri, 09 Dec 2022 08:31:25 GMT
x-cos-request-id: NjM5MmYyNWNfMzBkMDM4MGJfMmUzNzFfM****

<ObjectLockConfiguration>
  <ObjectLockEnabled>Enabled</ObjectLockEnabled>
  <Rule>
    <DefaultRetention>
      <Days>1</Days>
    </DefaultRetention>
  </Rule>
</ObjectLockConfiguration>
```

Querying the Object Locking Expiration Date

Description

This API is used to obtain the expiration date of object locking.

Request

Request Example

```
GET /<object-key>?retention HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> - <BucketName-APPID> is the bucket name with APPID as the suffix, for example, examplebucket-1250000000.
> - Authorization: Auth String (For details, refer to Request Signature).

Request Header

This API only uses common request headers. For details, please refer to the Common Request Headers document.

Request Body

The request body of this request is empty.

Response

Response Headers

This API only returns common response headers. For details, please refer to the Public Response Headers document.

Response Body

```
<?xml version="1.0" encoding="UTF-8" ?>
<Retention>
        <RetainUntilDate>timestamp</RetainUntilDate>
</Retention>
```

The detailed data is described as follows:

Node Name (Keyword)	Parent Node	Description	Type
Retention	No	Period	Container
RetainUntilDate	Retention	Specific expiration date	String Date

Error Code

This API follows unified error responses and error codes. For details, see Error Code.

Practical Case

Request

```
GET /temp.txt?retention HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Authorization: Auth String
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 87
Connection: keep-alive
Date: Fri, 09 Dec 2022 08:35:49 GMT
x-cos-request-id: NjM5MmYzNjVfMjBkMDM4MGJfMWM2ND****

<Retention>
   <RetainUntilDate>2022-12-10T08:34:48.000Z</RetainUntilDate>
</Retention>
```

Object APIs
Basic Operations
Uploading Objects in Simple Upload Mode

Description

The PUT Object API request allows uploading a local object (Object) to a specified bucket. This operation requires the requester to have write permissions for the bucket.

Notes

1. Write permissions for the bucket are required.
2. If the value of Content-Length in the request header is smaller than the actual length of data transmitted in the request body, COS still successfully creates the file. However, the object size is only equal to the size defined in Content-Length, and other data is discarded.
3. If the Object being added already exists with the same name, the newly uploaded file will overwrite the original file, and a successful response will return 200 OK.

Request

Request Example

```
PUT /<ObjectKey> HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (For details, please refer to Request Signature document).

Request Header

Common Headers

This request uses common request headers. For details, see Common Request Headers.

Non-common Headers

This operation's implementation can also use the following request headers.

Name	Description	Type	Required
Content-Disposition	File name defined in RFC 2616. It is saved as object metadata.	string	No
Content-Encoding	Encoding format defined in RFC 2616. It is saved as object metadata.	string	No
Expect	When using Expect: 100-continue, the request content is sent only after receiving confirmation from the server.	string	No
Expires	Cache policy defined in RFC 2616. It is saved as object metadata.	string	No
x-cos-meta-*	Including user-defined header suffixes and user-defined header information will be returned as Object metadata, with a size limit of 2 KB. Note: User-defined header information supports underscores, but user-defined header suffixes do not support underscores.	string	No
x-cos-storage-class	Sets the storage class of the Object, with enumeration values: STANDARD, STANDARD_IA, ARCHIVE. Default value: STANDARD	string	No
x-cos-acl	Defines the ACL property of the Object, with valid values: private, public-read, default; Default value: default (inherits Bucket permissions). Note: The current access policy entry limit is 1000 entries. If you do not need to control Object ACLs, please fill in default or leave this item unset, which will default to inheriting Bucket permissions.	string	No
x-cos-grant-read	Grants the grantee read permission, format: x-cos-grant-read: id="[OwnerUin]"	String	No
x-cos-grant-full-control	Grants the grantee full control permission, format: x-cos-grant-full-control: id="[OwnerUin]"	String	No

SSE-related headers

SSE can be used during object upload. For details, see Server-side encryption headers.

Request Body

The request body of this request is the content of the Object file.

Response

Response Headers

Common Response Headers

This response contains common response headers. For details on common response headers, please refer to the Common Response Header document.

Special Response Headers

Response headers for this request operation are as follows:

Name	Type	Description
ETag	string	MD5 value of the uploaded file content

Versioning-related headers

The following table describes the response headers returned when objects are uploaded to a versioning-enabled bucket.

Name	Description	Type
x-cos-version-id	Version ID of an object.	string

SSE-related headers

If SSE is used during object upload, this API returns headers used specifically for SSE. For details, see Server-Side Encryption Custom Header.

Response Body

The response body is empty.

Error Code

This request returns common error responses and error codes. For more information, see Error Codes.

Practical Case

Request

```
PUT /picture.jpg HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Wed, 28 Oct 2015 20:32:00 GMT
```

```
Authorization:q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84639384;32557535384&q-key-time=1484639384;32557535384&q-header-list=host&q-url-param-list
=&q-signature=5c07b7c67d56497d9aacb1adc19963135b7d00dc
Content-Length: 64

[Object]
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Date: Wed,16 Aug 2017 11: 59: 33 GMT
x-cos-request-id: NTk5NDMzYTRfMjQ4OGY3Xzc3NGRfMWY=
```

Example: Using Server Encryption SSE-C

Request

```
PUT /exampleobject HTTP/1.1
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Date: Fri, 10 Apr 2020 09:36:12 GMT
Content-Type: image/jpeg
x-cos-server-side-encryption-customer-algorithm: AES256
x-cos-server-side-encryption-customer-key: MDEyMzQ1Njc4OUFCQ0RFRjAxMjM0NTY3ODlBQkNERUY
=
x-cos-server-side-encryption-customer-key-MD5: U5L61r7jcwdNvT7frmUG8g==
Content-Length: 16
Content-MD5: 7o3pGNBWQBRbGPcPTDqmAg==
Authorization: q-sign-algorithm=sha1&q-ak=AKID8A0fBVtYFrNm02oY1g1JQQF0c3JO****&q-sign-tim
e=1586511372;1586518572&q-key-time=1586511372;1586518572&q-header-list=content-length;con
tent-md5;content-type;date;host;x-cos-server-side-encryption-customer-algorithm;x-cos-server-side-e
ncryption-customer-key;x-cos-server-side-encryption-customer-key-md5&q-url-param-list=&q-signat
ure=4f6f9f0a6700930f70bff31e3a2b2e622711****
Connection: close

[Object Content]
```

Response

```
HTTP/1.1 200 OK
Content-Length: 0
Connection: close
Date: Fri, 10 Apr 2020 09:36:13 GMT
ETag: "582d9105f71525f3c161984bc005efb5"
```

x-cos-hash-crc64ecma: 16749565679157681890
x-cos-request-id: NWU5MDNlMGNfZTFjODJhMDlfMzVlMDFfZTk1****
x-cos-server-side-encryption-customer-algorithm: AES256
x-cos-server-side-encryption-customer-key-MD5: U5L61r7jcwdNvT7frmUG8g==

Setting Object Replication

Description

The PUT Object - Copy is used to copy a file from the source path to the target path. The recommended file size is from 1 MB to 5 GB. For files larger than 5 GB, use the Upload - Copy API to perform the multipart upload. During the copy process, file meta-attributes and ACL can be modified.
Users can use this API to move files, rename files, modify file attributes, and create replicas.

Request

Request Example

```
PUT /<ObjectKey> HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
x-cos-copy-source: <BucketName-APPID>.<Endpoint>/filepath
```

> Note:
>
> Authorization: Auth String (For details, please refer to Request Signature section).

Request Header

Common Headers

This request uses common request headers. For details, see Public Request Headers.

Non-common Headers

Name	Description	Type	Required
x-cos-copy-source	Source file URL path. You can specify the historical version by using the versionid subresource.	string	Yes
x-cos-metadata-directive	Whether to copy the metadata of the source file, with enumerated values: Copy, Replaced, default value Copy. If marked as Copy, the metadata of the source file will be copied; if marked as Replaced, the metadata will be modified according to the Header information of this request. If the target path is the same as the source path, i.e., when the user attempts to modify the metadata, the mark must be	string	No

Name	Description	Type	Required
	Replaced.		
x-cos-copy-source-If-Modified-Since	If the object is modified after the specified time, the operation is performed; otherwise, 412 is returned. Can be used together with x-cos-copy-source-If-None-Match, conflicts with other conditions when used in combination.	string	No
x-cos-copy-source-If-Unmodified-Since	If the object has not been modified after the specified time, the operation is performed; otherwise, 412 is returned. Can be used together with x-cos-copy-source-If-Match, conflicts with other conditions when used in combination.	string	No
x-cos-copy-source-If-Match	When the Object's Etag is consistent with the given one, the operation is performed, otherwise 412 is returned. Can be used together with x-cos-copy-source-If-Unmodified-Since, conflicts with other conditions when used in combination.	string	No
x-cos-copy-source-If-None-Match	When the Etag of the Object is inconsistent with the given one, the operation is performed, otherwise 412 is returned. Can be used together with x-cos-copy-source-If-Modified-Since, conflicts with other conditions when used in combination.	string	No
x-cos-acl	Define the ACL attributes of an object. Valid values: private, public-read; default value: private	string	No
x-cos-grant-read	Grants read access to the grantee. Format: x-cos-grant-read: id="[OwnerUin]"	string	No
x-cos-grant-write	Grants write permission to the grantee. Format: x-cos-grant-write: id="[OwnerUin]"	string	No
x-cos-grant-full-control	Grants full permission to the grantee. Format: x-cos-grant-full-control: id="[OwnerUin]"	string	No
x-cos-meta-*	Includes user-defined header suffix and user-defined header information, which will be returned as Object metadata, with a size limit of 2 KB. Note: User-defined header information supports underscores, but user-defined header suffix does not support underscores.	string	No

Request Body

The request body of this request is empty.

Response

Response Headers

Common Response Headers

This response contains common response headers. For details on common response headers, please refer to the Common Response Header document.

Response Body

The response body returns application/xml data. The following contains all the nodes:

```
<CopyObjectResult>
  <ETag>"ba82b57cfdfda8bd17ad4e5879ebb4fe"</ETag>
  <LastModified>2017-08-04T02:41:45</LastModified>
</CopyObjectResult>
```

The data are as follows:

Name	Description	Type
CopyObjectResult	Returns the copy result information	String
ETag	Returns the MD5 algorithm check value of the file. The value of ETag can be used to check whether the object content is changed.	String
LastModified	Last modification time of the file, in GMT format.	String

Practical Case

Request

```
PUT /exampleobject HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Fri, 04 Aug 2017 02:41:45 GMT
Connection: keep-alive Accept-Encoding: gzip, deflate Accept: */*
User-Agent: python-requests/2.12.4
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
80932292;1981012292&q-key-time=1480932292;1981012292&q-url-param-list=&q-header-list=host
&q-signature=eacefe8e2a0dc8a18741d9a29707b1dfa5aa47cc
x-cos-copy-source: sourcebucket-1250000001.<Endpoint>/picture.jpg
Content-Length: 0
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 133
Connection: keep-alive
Date: Fri, 04 Aug 2017 02:41:45 GMT
x-cos-request-id: NTk4M2RlZTlfZDRiMDM1MGFfYTA1ZV8xMzNlYw==

<CopyObjectResult>
    <ETag>"ba82b57cfdfda8bd17ad4e5879ebb4fe"</ETag>
    <LastModified>2017-08-04T02:41:45</LastModified>
</CopyObjectResult>
```

Uploading Objects Through Forms

Description

The POST Object API is used to upload files (objects) to a specified bucket by using a form. This operation requires the requester to have the write permission on the bucket. All API parameters carried by HTTP headers are requested using form fields.

Notes

1. Write permissions for the bucket are required.
2. If the Object being added already exists with the same name, the newly uploaded file will overwrite the original file, and a successful response will return 200 OK.

Request

Request Example

```
POST / HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Content-Length: length
Headers
Form
```

Request Header

Common Headers

The implementation of this request operation uses common request headers. For details on common request headers, please refer to the Common Request Headers document.

Non-common Headers

The following table describes required request headers for this request operation.

Name	Description	Type	Required
Content-Length	Length (bytes) of HTTP request content as defined in RFC 2616.	String	Yes

Form Field

Name	Description	Type	Required
acl	Defines the ACL attribute of an object. Valid values: private, public-read, and default. Default value: default (inheriting bucket permissions). Note: Up to 1000 access policy entries are supported. If you do not need to set an ACL for the object, set this parameter to default or leave it empty. In this way, the object will inherit the permissions of the bucket it is stored in.	String	No
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	Header defined in RFC 2616. For details, see .	String	No
file	File content, which is used as the last field of the form.	String	Yes
key	Name of the file after upload. If ${filename} is used, the file name is replaced, for example, a/b/${filename}. If the photo.jpg file is uploaded, the final upload path is a/b/photo.jpg.	String	Yes
x-cos-meta-*	Includes user-defined header suffix and user-defined header information, which will be returned as Object metadata, with a size limit of 2 KB. Note: User-defined header information supports underscores. However, the user-defined header suffix does not support underscores.	String	No
policy	Base64 encoding, which is used for request check. If the request content does not match the conditions specified by Policy, 403 AccessDenied is returned.	String	No

Signature Protection

When an HTTP POST request for form upload is initiated and signature protection is required, the form should include form-data in the following content structure:

Form Field	Description
policy	Policy content encrypted using Base64, which will be used to check the request content. If the request content does not meet the specified conditions of the policy, the request will be rejected.

Form Field	Description
q-sign-algorithm	Algorithm used for calculating the signature. COS supports SHA1, and sha1 in lowercase is entered here.
q-ak	ID of the account key, that is, SecretId.
q-key-time	Effective start and end time of the key used for request signature, described by Unix timestamps, in seconds, in [start-seconds];[end-seconds] format, for example, 1480932292;1481012298 .
q-signature	Request signature calculated by using the preceding elements. COS verifies the form elements and signature content. If the signature does not match the signed content, the request is rejected.

Signature Calculation

The calculation of the signature q-signature is classified into three steps:

1. Use the key content to encrypt the value of q-key-time to obtain SignKey.
2. Create a POST request policy and encrypt the content using SHA1 to obtain StringToSign.
3. Use SignKey to encrypt StringToSign to generate the signature.

Policy

The following example is a complete policy.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"acl": "public-read" },
    {"bucket": "examplebucket-1250000000" },
    ["starts-with", "$key", "user/eric/"],
    {"q-sign-algorithm": "sha1" },
    {"q-ak": "AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" },
    {"q-sign-time": "1480932292;1481012298" }
  ]
}
```

Expiration

Sets the timeout time of POST Policy, in ISO8601 GMT time, for example, 2017-12-01T12:00:00.000Z.

Conditions rules

Type	Description
Exact match	Expressed using {"key": "value"} or ["eq", "$key", "value"]

Type	Description
Prefix match	Expressed using ["starts-with", "$key", "value"] , where the value can be left blank
Range match	Only used for ["content-length-range", int1, int2] , and the file byte count should be within the range of int1 to int2.

Conditions parameters

All parameters are optional. If a parameter is not specified, it does not need to be verified.

Name	Description	Matching Method
acl	ACL attribute of the object, which is optional.	Exact match or prefix match
bucket	Specified bucket to which data is uploaded.	Exact match
content-length-range	Specified size range of the uploaded file.	Scope
key	Storage path of an object.	Exact match or prefix match
x-cos-meta-*	Includes the user-defined header suffix and user-defined header information, which are returned as object metadata with a size limit of 2 KB. Note: User-defined header information supports underscores, but user-defined header suffix does not support underscores.	Exact match or prefix match
x-cos-*	Other COS headers that need to be signed.	Exact match

Response

Response Headers

Common Response Headers

This response uses common response headers. For details on common response headers, see Public Response Headers.

Response Parameters

Name	Description	Type

Name	Description	Type
ETag	Returns the MD5 checksum value of the file. The value of ETag can be used to check whether the object was corrupted during the upload process.	String

Response Body

The response body is returned as application/xml data, and the content containing the complete node data is shown below.

```
<PostResponse>
    <Location>http://examplebucket-1250000000.<Endpoint>/photo.jpg</Location>
    <Bucket>examplebucket-1250000000</Bucket>
    <Key>photo.jpg</Key>
    <ETag>d41d8cd98f00b204e9800998ecf8427e</ETag>
</PostResponse>
```

The data are described as follows:

Node Name (Keyword)	Parent Node	Description	Type	Required
PostResponse	No	Container for storing the results of the POST Object operation.	Container	Yes

Content of the PostResponse node of the container type:

Node Name (Keyword)	Parent Node	Description	Type	Required
Location	PostResponse	Full path of the object.	String	Yes
Bucket	PostResponse	Bucket where the object is located.	String	Yes
Key	PostResponse	Key name of the object.	String	Yes
ETag	PostResponse	Content of ETag.	String	Yes

Error Code

The following table describes some special and common errors that may occur with this request. For details on error codes related to COS, see Error Codes.

Error Code	HTTP Status Code	Description
InvalidDigest	400 Bad	If the user uploads a file with the Content-MD5 header, COS

Error Code	HTTP Status Code	Description
	Request	checks whether the Md5 in the body is consistent with the user-provided MD5. If not, COS returns an InvalidDigest error.
KeyTooLong	400 Bad Request	Custom header started with x-cos-meta carried while a file is uploaded. The key and value of each user-defined header cannot exceed 4 KB. Otherwise, a KeyTooLong error is returned.
MissingContentLength	411 Length Required	If the Content-Length header is not added when a file is uploaded, this error code is returned.
NoSuchBucket	404 Not Found	If the bucket containing the object you are trying to add does not exist, a 404 Not Found error is returned, and the error code is NoSuchBucket.
EntityTooLarge	400 Bad Request	If the size of the file to be added exceeds 5 GB, an EntityTooLarge error is returned, along with the message "Your proposed upload exceeds the maximum allowed object size" .
InvalidURI	400 Bad Request	The object key length is limited to 850 characters. If it exceeds 850 characters, an 'InvalidURI' error is returned.

Practical Case

Request

```
POST / HTTP/1.1
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: python-requests/2.12.4
Host: <BucketName-APPID>.<Endpoint>
Content-Length: 1352
Content-Type: multipart/form-data; boundary=e07f2a7876ae4755ae18d300807ad879

--e07f2a7876ae4755ae18d300807ad879
Content-Disposition: form-data; name="key"

a/${filename}
--e07f2a7876ae4755ae18d300807ad879
Content-Disposition: form-data; name="Acl"
```

public-read
--e07f2a7876ae4755ae18d300807ad879
Content-Disposition: form-data; name="Signature"

q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=1512983814;1512984814&q-key-time=1512983814;1512984814&q-url-param-list=&q-header-list=host&q-signature=2ffd2ae714e7445a8da000ec5d51771ff5056500
--e07f2a7876ae4755ae18d300807ad879
Content-Disposition: form-data; name="policy"

eyJjb25kaXRpb25zIjogW3siYnVja2V0IjogImtpdG1hbnMzdGVzdDEifSwgWyJjb250ZW50LWxlbmd0aC1yYW5nZSIsIDAsIDEwMDAwMDAwXSwgWyJzdGFydHMtd2l0aCIsICJ4LWNvcy1tZXRhLWJiIiwgIjEyIl1dLCAiZXhwaXJhdGlvbiI6ICIyMDQ3LTEyLTAxVDEyOjAwOjAwLjAwMFoifQ==
--e07f2a7876ae4755ae18d300807ad879
Content-Disposition: form-data; name="x-Cos-meta-bb"

124
--e07f2a7876ae4755ae18d300807ad879
Content-Disposition: form-data; name="key1"

1
--e07f2a7876ae4755ae18d300807ad879
Content-Disposition: form-data; name="file"; filename="empty:a"

--e07f2a7876ae4755ae18d300807ad879--

Response

HTTP/1.1 204
Content-Type: application/xml
Content-Length: 232
Connection: keep-alive
Date: Mon, 11 Dec 2017 09:16:56 GMT
ETag: "d41d8cd98f00b204e9800998ecf8427e"
Location: http://examplebucket-1250000000.cos.ap-[Region].myqcloud.com/photo.jpg
x-cos-request-id: NWEyZTRkMDZfMjQ4OGY3MGFfNTE4Yl81

Downloading an Object

Description

The GET Object API is used to download a file (object) from a COS bucket to a local directory. This operation requires the requester to have the read permission for the target object or the target object is publicly readable (public read).

Request

Request sample:

```
GET /<ObjectName> HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Line

```
GET /{ObjectName} HTTP/1.1
```

This API accepts GET ask.

Request Parameters

Name	Type	Required	Description
response-content-type	string	No	Set the Content-Type parameter in the response header.
response-content-language	string	No	Set the Content-Language parameter in the returned header.
response-expires	string	No	Set the Content-Expires parameter in the response header
response-cache-control	string	No	Set the Cache-Control parameter in the response header.

Name	Type	Required	Description
response-content-disposition	string	No	Set the Content-Disposition parameter in the response header.
response-content-encoding	string	No	Set the Content-Encoding parameter in the response header.
versionId	string	No	Version ID to download when versioning is enabled. If it is not specified, the latest version of the object is downloaded.

Request Header

Common Headers

This request uses common request headers. For details, see Common Request Headers.

Non-common Headers

Name	Type	Required	Description
Range	string	No	The specified file download range defined in RFC 2616, in bytes.
If-Unmodified-Since	string	No	The file content is returned only if the file modification time is earlier than or equal to the specified time. Otherwise, 412 (precondition failed) is returned.
If-Modified-Since	string	No	If the object has been modified after the specified time, the meta-information of the corresponding object is returned; otherwise, a 304 (not modified) response is returned.
If-Match	string	No	The file is returned only when the ETag matches the specified content. Otherwise, 412 (precondition failed) is returned.
If-None-Match	string	No	The file is returned only when the ETag does not match the specified content. Otherwise, 304 (not modified) is returned.

Request Body

The request does not have a request body.

Response

Response Headers

In addition to common response headers, this API returns the following response headers. For details about common response headers, see Common Response Headers.

Name	Description	Type
Cache-Control	Cache instruction defined in RFC 2616, which is returned only if it is contained in the object metadata or specified in the request parameters.	string
Content-Disposition	File name defined in RFC 2616, which is returned only if it is contained in the object metadata or specified in the request parameters.	string
Content-Encoding	Encoding format defined in RFC 2616, which is returned only if it is contained in the object metadata or specified in the request parameters.	string
Content-Range	Byte range of the content to be returned defined in RFC 2616, which is returned only when a Range request header is specified in the request.	string
Expires	Cache expiration time defined in RFC 2616, which is returned only if it is contained in the object metadata or specified in the request parameters.	string
x-cos-meta-*	Includes the user-defined metadata header suffix and user-defined metadata information.	string
x-cos-storage-class	Storage type of an object, such as STANDARD_IA and ARCHIVE. This header is returned only if the object is not of the standard storage (STANDARD) type.	Enum

Versioning-related headers

The following table describes the response headers returned for objects in a versioning-enabled bucket.

Name	Description	Type
x-cos-version-id	Version ID of an object.	string

Response Body

The download was successful and the file content is in the response body.

Error Code

Error Code	Description	HTTP Status Code
InvalidArgument	Wrong parameters provided	400 Bad Request

Error Code	Description	HTTP Status Code
SignatureDoesNotMatch	If this error code is returned, it means the provided signature is incorrect.	403 Forbidden
NoSuchKey	If the downloaded file does not exist, this error code is returned	404 Not Found

Practical Case

Request 1

```
GET /123 HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Wed, 28 Oct 2014 22:32:00 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84212200;32557108200&q-key-time=1484212200;32557108200&q-header-list=host&q-url-param-list
=&q-signature=11522aa3346819b7e5e841507d5b7f156f34e639
```

Response 1

```
HTTP/1.1 200 OK
Date: Wed, 28 Oct 2014 22:32:00 GMT
Content-Type: application/octet-stream
Content-Length: 16087
Connection: keep-alive
Accept-Ranges: bytes
Content-Disposition: attachment; filename=\"filename.jpg\"
Content-Range: bytes 0-16086/16087
ETag: \"9a4802d5c99dafe1c04da0a8e7e166bf\"
Last-Modified: Wed, 28 Oct 2014 20:30:00 GMT
x-cos-request-id: NTg3NzQ3ZmVfYmRjMzVfMzE5N182NzczMQ==

[Object]
```

Request 2

Carry response-xxx parameters

```
GET /123?response-content-type=application%2fxml HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Wed, 28 Oct 2014 22:32:00 GMT
```

Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=1484212200;32557108200&q-key-time=1484212200;32557108200&q-header-list=host&q-url-param-list=&q-signature=11522aa3346819b7e5e841507d5b7f156f34e639

Response 2

HTTP/1.1 200 OK
Date: Wed, 28 Oct 2014 22:32:00 GMT
Content-Type: application/xml
Content-Length: 16087
Connection: keep-alive
Accept-Ranges: bytes
Content-Disposition: attachment; filename=\"filename.jpg\"
Content-Range: bytes 0-16086/16087
ETag: \"9a4802d5c99dafe1c04da0a8e7e166bf\"
Last-Modified: Wed, 28 Oct 2014 20:30:00 GMT
x-cos-request-id: NTg3NzQ3ZmVfYmRjMzVfMzE5N182NzczMQ==

[Object]

Querying Metadata of Objects

Description

The HEAD Object API request allows fetching meta-information data for the corresponding object. The permissions for HEAD are the same as those for GET.

Request

Request sample:

```
HEAD /<ObjectKey> HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (For details, please refer to the Request Signature document).

Request Header

Common Headers

This request uses common request headers. For details, see Common Request Headers.

Non-common Headers

Name	Type	Required	Description
If-Modified-Since	string	No	If the object has been modified after the specified time, the meta-information of the corresponding object is returned; otherwise, a 304 response is returned.

Request Body

The request does not have a request body.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Common Response Headers.

Special Response Headers

Response headers for this request operation are as follows:

Name	Type	Description
x-cos-meta- *	string	User-defined metadata

Response Body

The response body is empty.

Practical Case

Request

```
HEAD /exampleobject HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Thu, 12 Jan 2017 17:26:53 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84213210;32557109210&q-key-time=1484213210;32557109210&q-header-list=host&q-url-param-list
=&q-signature=ac61b8eb61964e7e6b935e89de163a479a25c210
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 16087
Connection: keep-alive
Date: Thu, 12 Jan 2017 17:26:53 GMT
ETag: \"9a4802d5c99dafe1c04da0a8e7e166bf\"
Last-Modified: Wed, 11 Jan 2017 07:30:07 GMT
x-cos-request-id: NTg3NzRiZGRfYmRjMzVfM2Y2OF81N2YzNA==
```

Deleting an Object

Description

The Delete Object API is used to delete a file (object) in a COS bucket. This operation requires the requester to have the write permission on the bucket.

Notes

1. In a Delete Object request, deleting a non-existing object is still considered successful, and it returns 204 No Content .
2. Delete Object requires the user to have write permissions for that object.

Versioning

To delete a specific version of an object (including the delete marker, same as below), use the versionId parameter to specify the version ID (including the version ID of the delete marker, same as below). In this case, the response includes the x-cos-version-id response header, representing the version ID deleted by this request.

If the versionId parameter is not specified.

- When versioning is enabled, the DELETE operation creates a delete marker as the latest version of the specified object. The response returns the x-cos-version-id response header, representing the version ID of the delete marker created by this request.
- When versioning is suspended, the DELETE operation creates a delete marker whose version ID is null as the latest version of the specified object and deletes any existing versions with a null version ID (if any).

When the DELETE operation creates or deletes a delete marker, the x-cos-delete-marker: true response header is returned, indicating that the DELETE operation has created or deleted the delete marker of the specified object.
For details about the status (enabled/suspended) of versioning, see Version Control Overview.

Request

Syntax example:

```
DELETE /ObjectName HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
```

Date: GMT Date
Content-Length: length
Authorization: Auth String

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Line

```
DELETE /ObjectName HTTP/1.1
```

This API accepts DELETE requests.

Request Parameters

Name	Description	Type	Required
versionId	Version ID to be deleted.	string	No

Request Header

Common Headers

This request uses common request headers. For details, see Common Request Headers.

Non-common Headers

This request has no special request header information.

Request Body

The request body for this request is empty.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Public Response Header.

Special Response Headers

There are no special response headers for this request operation.

Versioning-related headers

The following table describes response headers returned when you delete an object or a specified object

version from a versioning-enabled bucket.

Name	Description	Type
x-cos-version-id	Version ID of the object or delete marker.	string
x-cos-delete-marker	- When the versionId parameter is used to specify the version ID of a delete marker, this response header is returned with a value of true, indicating that the version ID of a delete marker is deleted. - When the versionId parameter is not used, and versioning is enabled for the bucket where the specified object is located, this response header is returned with a value of true, indicating that a delete marker is created as the latest version of the object.	boolean

Response Body

The response body of this request is empty.

Error Analysis

The following describes some special but common errors that may occur with this request:

Error Code	HTTP Status Code	Description
NoSuchBucket	404 Not Found	The bucket does not exist.

For details on COS error codes or a complete list of product error codes, see Error Codes.

Practical Case

Request

```
DELETE /123 HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Wed, 23 Oct 2016 21:32:00 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84213409;32557109409&q-key-time=1484213409;32557109409&q-header-list=host&q-url-param-list
=&q-signature=1c24fe260ffe79b8603f932c4e916a6cbb0af44a
```

Response

```
HTTP/1.1 204 No Content
Content-Type: application/xml
```

```
Content-Length: 0
Connection: keep-alive
Date: Wed, 23 Oct 2016 21:32:00 GMT
x-cos-request-id: NTg3NzRjYTRfYmRjMzVfMzFhOF82MmM3Yg==
```

Deleting Multiple Objects

Description

The DELETE Multiple Object API is used to delete objects in a specified bucket in batches. Up to 1000 objects can be deleted at a time. COS offers two response modes: Verbose and Quiet. In Verbose mode, the deletion result of each object is returned. In Quiet mode, only information about objects that had errors is returned.

> Note:
>
> This request must carry Content-MD5 to verify the integrity of the Body.

Notes

1. Each batch delete request can contain a maximum of 1,000 objects to be deleted.
2. Batch deletion supports two modes of return: verbose and quiet. The default mode is verbose. The verbose mode returns the deletion status of each key, and the quiet mode only returns the status of the keys that failed to be deleted.
3. Batch deletion requires the Content-MD5 header to verify that the request body has not been modified.
4. Batch deletion requests allow a non-existent key to be deleted and still be considered successful.

Versioning

If versioning is enabled, you can specify a version ID for each object to delete, and the specified version or delete marker is permanently deleted. Otherwise, a new delete marker is created as the latest version of the specified object.

If a delete marker is created or deleted for an object deletion operation, both true and are included in the deletion result of the object, indicating that the delete marker of the specified object is created or deleted.

If a specific version ID (including the version ID of the delete marker) is permanently deleted in an object deletion operation, the deletion result includes the element, indicating the version ID that was deleted.

Request

Syntax example:

```
POST /?delete HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
```

```
Date: GMT Date
Content-Length: length
Content-Type: application/xml
Content-MD5: MD5
Authorization: Auth String

<Delete>
  <Quiet></Quiet>
  <Object>
    <Key></Key>
  </Object>
  <Object>
    <Key></Key>
  </Object>
  ...
</Delete>
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Line

```
POST /?delete HTTP/1.1
```

This API accepts POST requests.

Request Header

Common Headers

This request operation is implemented by using the public request header. For details about the public request header, see Common Request Headers documentation.

Non-common Headers

Required headers:
This request is implemented by using the following required headers.

Name	Description	Type	Required
Content-Length	HTTP request content length (bytes) defined in RFC 2616.	String	Yes
Content-MD5	Base64-encoded 128-bit content MD5 checksum as defined in RFC 1864. This header is used to verify whether the file content has changed.	String	Yes

Request Body

The specific node content of the request body of this request is as follows.

```
<Delete>
  <Quiet></Quiet>
  <Object>
    <Key></Key>
  </Object>
  <Object>
    <Key></Key>
  </Object>
  ...
</Delete>
```

The specific contents are described as follows:

Node Name (Keyword)	Parent Node	Description	Type	Required
DELETE	No	Describes the return result method and target object of this deletion.	Container	Yes
Quiet	DELETE	Boolean value, this value determines whether to enable the Quiet mode. A value of true starts Quiet mode, a value of false starts Verbose mode, and the default value is False.	Boolean	No
Object	DELETE	Describes the information of each target object to be deleted.	Container	Yes
Key	DELETE.Object	Target Object File Name	String	Yes

Response

Response Headers

Common Response Headers

This response utilizes common response headers. For details on common response headers, please refer to the Public Response Header document.

Special Response Headers

There are no special response headers for this request operation.

Response Body

The response body returns application/xml data. The following contains all the nodes:

```
<DeleteResult>
  <Deleted>
    <Key></Key>
  </Deleted>
  <Error>
    <Key></Key>
    <Code></Code>
    <Message></Message>
  </Error>
</DeleteResult>
```

The specific contents are as follows:

Node Name (Keyword)	Parent Node	Description	Type
DeleteResult	No	Describes the return result method and target object of this deletion.	Container

Contents of the Container node DeleteResult:

Node Name (Keyword)	Parent Node	Description	Type
Deleted	DeleteResult	Information about objects that are successfully deleted.	Boolean
Error	DeleteResult	Information about objects that fail to be deleted.	Container

Contents of the Container node Deleted:

Node Name (Keyword)	Parent Node	Description	Type
Key	DeleteResult.Deleted	Name of an object.	String

Contents of the Container node Error:

Node Name (Keyword)	Parent Node	Description	Type
Key	DeleteResult.Error	The name of the object that failed to be	String

Node Name (Keyword)	Parent Node	Description	Type
		deleted.	
Code	DeleteResult.Error	Deletion failed error code	String
Message	DeleteResult.Error	Deletion failed error message	String

Error Analysis

The following describes some special but common errors that may occur with this request:

Error Code	HTTP Status Code	Description
InvalidRequest	400 Bad Request	The required field Content-MD5 is not carried, and error message Missing required header for this request: Content-MD5 is returned.
MalformedXML	400 Bad Request	If the number of requested keys exceeds 1000, a MalformedXML error will be returned, along with delete key size being greater than 1000 .
InvalidDigest	400 Bad Request	The Content-MD5 carried is inconsistent with the request body calculated by the server.

For more information on COS error codes or the complete list of product errors, please refer to the Error Codes document.

Practical Case

Request

```
POST /?delete HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Wed, 23 Oct 2016 21:32:00 GMT
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: python-requests/2.12.4
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
80932292;1981012292&q-key-time=1480932292;1981012292&q-url-param-list=delete&q-header-list
=host&q-signature=c54f22fd92232a76972ba599cba25a8a733d2fef
Content-MD5: yoLiNjQuvB7lu8cEmPafrQ==
```

```
Content-Length: 125

<Delete>
  <Quiet>true</Quiet>
  <Object>
    <Key>aa</Key>
  </Object>
  <Object>
    <Key>aaa</Key>
  </Object>
</Delete>
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 17
Connection: keep-alive
Date: Tue, 22 Aug 2017 12:00:48 GMT
x-cos-request-id: NTk5YzFjZjBfZWFhZDM1MGFfMjkwZV9lZGM3ZQ==

<DeleteResult/>
```

Request

```
POST /?delete HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Tue, 22 Aug 2017 12:16:35 GMT
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: python-requests/2.12.4
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
80932292;1981012292&q-key-time=1480932292;1981012292&q-url-param-list=delete&q-header-list
=host&q-signature=c54f22fd92232a76972ba599cba25a8a733d2fef
Content-MD5: V0XuU8V7aqMYeWyD3BC2nQ==
Content-Length: 126

<Delete>
  <Quiet>false</Quiet>
  <Object>
    <Key>aa</Key>
  </Object>
  <Object>
    <Key>aaa</Key>
```

```
  </Object>
</Delete>
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 111
Connection: keep-alive
Date: Tue, 22 Aug 2017 12:16:35 GMT
x-cos-request-id: NTk5YzIwYTNfMzFhYzM1MGFfMmNmOWZfZWVhNjQ=

<DeleteResult>
 <Deleted>
  <Key>aa</Key>
 </Deleted>
 <Deleted>
  <Key>aaa</Key>
 </Deleted>
</DeleteResult>
```

Checking CORS Configuration

Description

The OPTIONS Object API is used to send an OPTIONS request before the CORS configuration for an object. That is, an OPTIONS request with the specific source domain, HTTP method, and header information is sent to COS before a CORS configuration request to determine whether a CORS configuration request can be sent. When the CORS configuration does not exist, a 403 Forbidden response is returned. You can call the PUT Bucket cors API to enable CORS for the bucket.

Request

Request Example

```
OPTIONS /<ObjectKey> HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Origin: Origin
Access-Control-Request-Method: HTTPMethod
Access-Control-Request-Headers: RequestHeader
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (For details, please refer to section Request Signature).

Request Header

Common Headers

This request uses common request headers. For details, see Public Request Headers.

Non-common Headers

Name	Type	Description	Required
Origin	string	The domain name of the request source for simulating cross-origin access	Yes
Access-Control-Request-Method	string	The HTTP method for simulating cross-origin access requests.	Yes

Name	Type	Description	Required
Access-Control-Request-Headers	string	Request headers for simulating the cross-origin access	No

Request Body

The request body of this request is empty.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Common Response Headers.

Special Response Headers

Specific response headers for this request operation are as follows:

Name	Type	Description
Access-Control-Allow-Origin	string	Simulated cross-origin access request origin domain. This header is not returned when the origin is not allowed.
Access-Control-Allow-Methods	string	Simulated cross-origin access request HTTP method. This header is not returned when the request method is not allowed.
Access-Control-Allow-Headers	string	Simulated cross-origin access request headers. This header is not returned when simulating any request headers is not allowed.
Access-Control-Expose-Headers	string	Simulated cross-origin access request HTTP method. This header is not returned when the request method is not allowed.
Access-Control-Max-Age	string	Validity period of the result obtained by the OPTIONS request.

Response Body

The response body is empty.

Practical Case

Request

```
OPTIONS /exampleobject HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Thu, 12 Jan 2017 17:26:53 GMT
Origin: http://www.domain.com
Access-Control-Request-Method: PUT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
87070734;32466654734&q-key-time=1487070734;32559966734&q-header-list=host&q-url-param-list
=&q-signature=2ac3ada19910f44836ae0df72a0ec1003f34324b
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 16087
Connection: keep-alive
x-cos-request-id: NTg3NzRiZGRfYmRjMzVfM2Y2OF81N2YzNA==
Date: Thu, 12 Jan 2017 17:26:53 GMT
ETag: \"9a4802d5c99dafe1c04da0a8e7e166bf\"
Access-Control-Allow-Origin: http://www.domain.com
Access-Control-Allow-Methods: PUT
Access-Control-Expose-Headers: x-cos-request-id
```

Recovering Archived Objects

Description

The POST Object restore API can be used to restore an object that is archived as the archive storage type by COS. The restored readable object is temporary, and you can set the duration for it to remain readable as well as specify when to delete this temporary replica. You can use the Days parameter to set the expiration time of the temporary object. If it exceeds this duration and you have not initiated any copy, extension, or other operations, the temporary object will be automatically deleted by the system. The temporary object is only a replica of the object of the archive storage type, and the archived source object always exists during this period.

Request

Request Example

```
POST /<ObjectKey>?restore HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

Request Header

Common Headers

Public request headers are used for implementing this request.

Non-common Headers

This request has no special request header information.

Request Body

The following figure shows the request bodies required for implementing this request operation.

```
<RestoreRequest>
  <Days>2</Days>
  <CASJobParameters>
     <Tier>Bulk</Tier>
  </CASJobParameters>
</RestoreRequest>
```

The data are described as follows:

Node Name (Keyword)	Parent Node	Description	Type	Required
RestoreRequest	No	Container for data restoration.	Container	Yes
Days	No	Expiration time of the temporary replica.	integer	Yes
CASJobParameters	No	Restores job parameters.	Container	Yes
Tier	No	During data recovery, Tier can be specified as one of the three recovery modes supported by CAS: Standard (standard mode, with the recovery task completed within 3–5 hours), Expedited (speed mode, with the recovery task completed within 15 minutes), and Bulk (batch mode, with the recovery task completed within 5–12 hours).	Enum	Yes

Response

Response Headers

Common Response Headers

This response contains common response headers.

Special Response Headers

There are no special response headers for this response operation.

Response Body

The response body is empty.

Error Code

The following table describes errors that may occur with this request.

Error Code	Description	HTTP Status Code
None	Restored successfully.	202 Accepted
RestoreAlreadyInProgress	Restoring the object.	409 Conflict

Practical Case

Request

```
POST /exampleobject?restore HTTP/1.1
Accept: */*
Authorization:q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
97530202;1497610202&q-key-time=1497530202;1497610202&q-header-list=&q-url-param-list=&q-s
ignature=28e9a4986df11bed0255e97ff90500557e0ea057
Host: <BucketName-APPID>.<Endpoint>
Content-Length: 105
Content-Type: application/x-www-form-urlencoded

<RestoreRequest>
  <Days>2</Days>
  <CASJobParameters>
     <Tier>Bulk</Tier>
  </CASJobParameters>
</RestoreRequest>
```

Response

```
HTTP/1.1 202 Accepted
Content-Type: application/xml
Content-Length: 0
Connection: keep-alive
Date: Thu, 15 Jun 2017 12:37:29 GMT
x-cos-request-id: NTk0MjdmODlfMjQ4OGY3XzYzYzhfMjc=
```

Access Control
Setting an ACL for an object

Description

The PUT Object acl API is used to configure the ACL table for an Object in a Bucket. You can pass ACL information through the Header: "x-cos-acl", "x-cos-grant-read", "x-cos-grant-full-control", or pass ACL information through the Body in XML format.

Request

Request Example

```
PUT /<ObjectKey>?acl HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (For details, please refer to the Request Signature document).

Request Header

Common Headers

This request uses common request headers. For details, see Common Request Headers.

Non-common Headers

Name	Description	Type	Required
x-cos-acl	Define the ACL property of the Object, with valid values: private, public-read, default; Default value: default (inherits Bucket permissions). Note: The current access policy entry limit is 1000. If you do not need to control Object ACL, please fill in default or leave this item unset, inheriting Bucket permissions by default.	String	No
x-cos-grant-read	Grants read access to the grantee. Format: x-cos-grant-read: id="[OwnerUin]"	String	No
x-cos-grant-full-control	Grants full permission to the grantee. Format: x-cos-grant-full-control: id="[OwnerUin]"	String	No

Request Body

The response body returns application/xml data. The following contains all the nodes:

```
<AccessControlPolicy>
  <Owner>
    <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
    <DisplayName>qcs::cam::uin/100000000001:uin/100000000001</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="CanonicalUser">
        <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
        <DisplayName>qcs::cam::uin/100000000001:uin/100000000001</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="CanonicalUser">
        <ID>qcs::cam::uin/100000000002:uin/100000000002</ID>
        <DisplayName>qcs::cam::uin/100000000002:uin/100000000002</DisplayName>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

The data are as follows:

Node Name (Keyword)	Parent Node	Description	Type	Required
AccessControlPolicy	No	Container for storing the results of the GET Object ACL	Container	Yes

Content of AccessControlPolicy:

Node Name (Keyword)	Parent Node	Description	Type	Required
Owner	AccessControlPolicy	Object Owner Information	Container	Yes
AccessControlList	AccessControlPolicy	Information about the grantee and permissions	Container	Yes

Content of Owner:

Node Name (Keyword)	Parent Node	Description	Type	Required
ID	AccessControlPolicy.Owner	Object Owner ID, Format: qcs::cam::uin/:uin/ If the ID is that of the root account, and have the same value.	String	Yes

Node Name (Keyword)	Parent Node	Description	Type	Required
DisplayName	AccessControlPolicy.Owner	Object Owner's Name	String	Yes

Content of AccessControlList:

Node Name (Keyword)	Parent Node	Description	Type	Required
Grant	AccessControlPolicy.AccessControlList	Authorization information for a single object. An AccessControlList can have 100 grants.	Container	Yes

Contents of Grant:

Node Name (Keyword)	Parent Node	Description	Type	Required
Grantee	AccessControlPolicy.AccessControlList.Grant	Grantee Information. The type can be RootAccount or Subaccount. When the type is RootAccount, the ID specifies the root account; when the type is Subaccount, the ID specifies the sub-account.	Container	Yes
Permission	AccessControlPolicy.AccessControlList.Grant	Specifies the permissions granted to the grantee, with enum values: READ, FULL_CONTROL.	String	Yes

Contents of Grantee:

Node Name (Keyword)	Parent Node	Description	Type	Required
URI	AccessControlPolicy.AccessControlList.Grant.Grantee	Specifies all users.	String	Yes
ID	AccessControlPolicy.AccessControlList.Grant.Grantee	User ID, formatted as: qcs::cam::uin/:uin/. If it is the root account, and have the same value.	String	Yes
DisplayName	AccessControlPolicy.AccessControlList.Grant.Grantee	User name	String	Yes

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Common Response Headers.

Special Response Headers

This request has no special response header information.

Response Body

The response body is empty.

Error Code

The response may contain the following error code information. For common error messages, please refer to the Error Codes document.

Error Code	Description	Status Code
SignatureDoesNotMatch	If this error code is returned, it means the provided signature is incorrect.	403 Forbidden
NoSuchBucket	If this error code is returned, it means the bucket you want to add the rule to is not found.	404 Not Found
MalformedXML	Invalid XML format. Compare it carefully with the RESTful API documentation.	400 Bad Request
InvalidRequest	The request is invalid. If the error description shows "header acl and body acl conflict," it means that ACL parameters cannot be present in both the header and body.	400 Bad Request

Practical Case

Request

```
PUT /exampleobject?acl HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Fri, 25 Feb 2017 04:10:22 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=1484724784;32557620784&q-key-time=1484724784;32557620784&q-header-list=host&q-url-param-list=acl&q-signature=785d9075b8154119e6a075713c1b9e56ff0bddfc
Content-Length: 229
Content-Type: application/x-www-form-urlencoded

<AccessControlPolicy>
  <Owner>
    <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
```

```
  <DisplayName>qcs::cam::uin/100000000001:uin/100000000001</DisplayName>
 </Owner>
 <AccessControlList>
  <Grant>
   <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="CanonicalUser">
    <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
    <DisplayName>qcs::cam::uin/100000000001:uin/100000000001</DisplayName>
   </Grantee>
   <Permission>FULL_CONTROL</Permission>
  </Grant>
  <Grant>
   <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="CanonicalUser">
    <ID>qcs::cam::uin/100000000002:uin/100000000002</ID>
    <DisplayName>qcs::cam::uin/100000000002:uin/100000000002</DisplayName>
   </Grantee>
   <Permission>READ</Permission>
  </Grant>
 </AccessControlList>
</AccessControlPolicy>
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Connection: keep-alive
Date: Fri, 25 Feb 2017 04:10:22 GMT\

x-cos-request-id: NTg3ZjFjMmJfOWIxZjRlXzZmNDhfMjIw
```

Querying the ACL of an Object

Description

The GET Object acl API is used to retrieve the access permissions for a specific object under a particular bucket, with access only granted to the bucket owner.

Request

Request Example

```
GET /<ObjectKey>?acl HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (For more details, please refer to Request Signature documentation).

Request Header

Common Headers

This request uses common request headers. For details, see Common Request Headers.

Non-common Headers

Required headers:
This request is implemented by using the following required headers.

Name	Description	Type	Required
Authorization	Signature String	String	Yes

Request Body

The request body of this request is empty.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Common Response Headers.

Special Response Headers

There are no special response headers for this response operation.

Response Body

The response body returns application/xml data. The following contains all the nodes:

```
<AccessControlPolicy>
  <Owner>
    <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
    <DisplayName>qcs::cam::uin/100000000001:uin/100000000001</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="CanonicalUser">
        <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
        <DisplayName>qcs::cam::uin/100000000001:uin/100000000001</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="CanonicalUser">
        <ID>qcs::cam::uin/100000000002:uin/100000000002</ID>
        <DisplayName>qcs::cam::uin/100000000002:uin/100000000002</DisplayName>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

The data are as follows:

Node Name (Keyword)	Parent Node	Description	Type
AccessControlPolicy	No	Container for storing the results of the GET Object ACL	Container

Content of AccessControlPolicy:

Node Name (Keyword)	Parent Node	Description	Type
Owner	AccessControlPolicy	Object Owner Information	Container
AccessControlList	AccessControlPolicy	Information about the grantee and permissions	Container

Content of Owner:

Node Name (Keyword)	Parent Node	Description	Type
ID	AccessControlPolicy.Owner	Object Owner ID, Formatted as: qcs::cam::uin/:uin/. If it's the root account, and have the same value.	String
DisplayName	AccessControlPolicy.Owner	Object Owner's Name	String

Content of AccessControlList:

Node Name (Keyword)	Parent Node	Description	Type
Grant	AccessControlPolicy.AccessControlList	Authorization information for a single object. An AccessControlList can have up to 100 Grant entries.	Container

Contents of Grant:

Node Name (Keyword)	Parent Node	Description	Type
Grantee	AccessControlPolicy.AccessControlList.Grant	Grantee Information. The type can be RootAccount or Subaccount. When the type is RootAccount, the ID specifies the root account; when the type is Subaccount, the ID specifies the sub-account.	Container

Node Name (Keyword)	Parent Node	Description	Type
Permission	AccessControlPolicy.AccessControlList.Grant	Specifies the permissions granted to the grantee, with enum values: READ, FULL_CONTROL.	String

Contents of Grantee:

Node Name (Keyword)	Parent Node	Description	Type
URI	AccessControlPolicy.AccessControlList.Grant.Grantee	Specifies all users.	String
ID	AccessControlPolicy.AccessControlList.Grant.Grantee	User ID, formatted as: qcs::cam::uin/:uin/. If it is a root account, and have the same value.	String
DisplayName	AccessControlPolicy.AccessControlList.Grant.Grantee	User name	String

Practical Case

Request

```
GET /exampleobject?acl HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Fri, 10 Mar 2016 09:45:46 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxx&q-sign-time=1484213027;32557109
027&q-key-time=1484213027;32557109027&q-header-list=host&q-url-param-list=acl&q-signature=
dcc1eb2022b79cb2a780bf062d3a40e120b4065
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 266
Connection: keep-alive
Date: Fri, 10 Mar 2016 09:45:46 GMT
```

x-cos-request-id: NTg3NzRiMjVfYmRjMzVfMTViMl82ZGZmNw==

```
<AccessControlPolicy>
  <Owner>
    <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
    <DisplayName>qcs::cam::uin/100000000001:uin/100000000001</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="CanonicalUser">
        <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
        <DisplayName>qcs::cam::uin/100000000001:uin/100000000001</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="CanonicalUser">
        <ID>qcs::cam::uin/100000000002:uin/100000000002</ID>
        <DisplayName>qcs::cam::uin/100000000002:uin/100000000002</DisplayName>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

Object Tag
Setting Object Tags

Description

COS allows you to set tags for existing objects. The PUT Object tagging API is used to add key-value pairs for objects as object tags to help you group and manage existing object resources. For details, see Object Tag Overview.
If you use a sub-account to call this interface, make sure you have obtained permission for PUT Object tagging at the root account level.

> Note
>
> - The object tagging feature can set up to 10 different tags for a single object.
> - If object tags already exist, calling this API will overwrite the existing tags.

Versioning

If versioning is enabled for your bucket and you need to add tags to a specific version of an object, you can include the VersionId parameter in the request, and the object tags will be added to the specified object version.

Request

Request Example

```
PUT /<ObjectKey>?tagging&VersionId=VersionId HTTP 1.1
Host:<BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Description
>
> - indicates the bucket name with APPID as the suffix, for example, examplebucket-1250000000.
> - Authorization: Auth String (For details, please refer to the Request Signature document).

Request Parameters

Name	Description	Type	Required
versionId	When versioning is enabled, specify the version ID of the object to	string	No

Name	Description	Type	Required
	operate on. If it is not specified, the tags are added to the latest version of the object.		

Request Header

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

This request needs to set the following tag set:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Tagging>
  <TagSet>
    <Tag>
      <Key>string</Key>
      <Value>string</Value>
    </Tag>
  </TagSet>
</Tagging>
```

The data are described as follows:

Node Name (Keyword)	Parent Node	Description	Type	Required
Tagging	No	Tag Set	Container	Yes
TagSet	Tagging	Tag Set	Container	Yes
Tag	Tagging.TagSet	Tag set, supports up to 10 tags.	Containers	Yes
Key	Tagging.TagSet.Tag	Tag key, which cannot be empty and has a length of less than 128 bytes, supporting letters, digits, spaces, plus signs, minus signs, underscores, equal signs, periods, colons, and slashes.	String	Yes
Value	Tagging.TagSet.Tag	Tag value, with a length of less than 256 bytes, supporting letters, digits, spaces, plus signs, minus signs, underscores, equal signs, periods, colons, and slashes.	String	Yes

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

The response body is empty.

Error Code

This API follows unified error response and error codes. For details, please refer to the Error Codes document.

Practical Case

Request

The following request adds the {age:18} and {name:xiaoming} tags to the exampleobject.txt object in the examplebucket-1250000000 bucket. COS configures the tags and returns 200 (success).

```
PUT /exampleobject.txt?tagging HTTP/1.1
User-Agent: curl/7.29.0
Accept: */*
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Authorization: Auth String
Content-Length: 127
Content-MD5:MD5 String
Content-Type: application/xml

<Tagging>
  <TagSet>
    <Tag>
      <Key>age</Key>
      <Value>18</Value>
    </Tag>
    <Tag>
      <Key>name</Key>
      <Value>xiaoming</Value>
    </Tag>
  </TagSet>
</Tagging>
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Connection: keep-alive
Date: Fri, 19 Jan 2020 11:40:22 GMT
x-cos-request-id: NWE2MWQ5MjZfMTBhYzM1MGFfMTA5ODVfMTVj****
```

Querying Object Tags

Description

The GET Object tagging API is used to query existing tags of a specified object.
If you are using a sub-account to call this API, please make sure that you have obtained permission for the GET Object tagging API at the root account.

Versioning

If your bucket has versioning enabled and you need to query the tags of a specified version of the object, you can include the VersionId parameter when making the request, which will then query the tags of the specified version object.

Request

Request Example

```
GET /<ObjectKey>?tagging&VersionId=VersionId HTTP 1.1
Host:<BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Description
>
> - indicates the bucket name with APPID as the suffix, for example, examplebucket-1250000000.
> - Authorization: Auth String (For details, refer to Request Signature).

Request Parameters

Name	Description	Type	Required
versionId	When versioning is enabled, specify the object VersionId to operate on. If not specified, the object tags of the latest version will be queried.	string	No

Request Header

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

The request body of this request is empty.

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

Description of the response body nodes returned by this request:

Node Name (Keyword)	Parent Node	Description	Type
Tagging	No	Tag Set	Container
TagSet	Tagging	Tag Set	Container
Tag	Tagging.TagSet	Tag set, supporting up to 10 tags	Containers
Key	Tagging.TagSet.Tag	Tag key, with a length of less than 128 bytes, supporting letters, digits, spaces, plus signs, minus signs, underscores, equal signs, periods, colons, and slashes.	String
Value	Tagging.TagSet.Tag	Tag value, with a length of less than 256 bytes, supporting letters, digits, spaces, plus signs, minus signs, underscores, equal signs, periods, colons, and slashes.	String

Error Code

This API follows unified error response and error codes. For details, please refer to the Error Codes document.

Practical Case

Request

The following request queries the tags of the object exampleobject.txt in the bucket examplebucket-1250000000 . After parsing the request, COS returns the tags {age:18} and {name:xiaoming} in the bucket.

```
GET /exampleobject.txt?tagging HTTP/1.1
User-Agent: curl/7.29.0
Accept: */*
```

```
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Authorization: Auth String
Content-Md5: MD5 String
Content-Length: 127
Content-Type: application/xml
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Connection: close
Date: Fri, 19 Jan 2020 11:40:22 GMT

<Tagging>
   <TagSet>
      <Tag>
         <Key>age</Key>
         <Value>18</Value>
      </Tag>
      <Tag>
         <Key>name</Key>
         <Value>xiaoming</Value>
      </Tag>
   </TagSet>
</Tagging>
```

Deleting Object Tags

Description

The DELETE Object tagging API is used to delete existing object tags for a specified object.
If you use a sub-account to call this API, please make sure you have obtained the permission for DELETE Object tagging API at the root account.
Version Control
If your bucket has versioning enabled and you need to delete tags for a specific versioned object, you can include the VersionId parameter in the request. This will delete the object tags set for the specified version ID.

Request

Request Example

```
DELETE /<ObjecKey>?tagging&VersionId=VersionId HTTP 1.1
Host:<BucketName-APPID>.<Endpoint>
Date:date
Authorization: Auth String
```

> Description
>
> - indicates the bucket name with APPID as the suffix, for example, examplebucket-1250000000.
> - Authorization: Auth String (For details, refer to Request Signature).

Request Parameters

Name	Description	Type	Required
versionId	When versioning is enabled, specify the version ID of the object to be manipulated. If not specified, the tags of the latest version of the object will be deleted.	string	No

Request Header

This API only uses common request headers. For details, see Common Request Headers documentation.

Request Body

The request body of this request is empty.

Response

Response Headers

This API only returns the public response header. For details, see Common Response Headers documentation.

Response Body

This request does not have any specific response body information.

Error Code

This API follows unified error response and error codes. For details, please refer to the Error Codes document.

Practical Case

Case One: Deleting Object Tags (Versioning Not Enabled)

Request

The following request deletes existing tags from the object exampleobject.txt in the bucket examplebucket-1250000000 . After parsing this request, COS deletes all tags in the object.

```
DELETE /exampleobject.txt?tagging HTTP/1.1
User-Agent: curl/7.29.0
Accept: */*
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Authorization: Auth String
Content-Length: 0
Content-Type: application/xml
```

Response

```
HTTP/1.1 204 No Content
Content-Type: application/xml
Connection: close
Date: Fri, 19 Jan 2020 11:40:22 GMT
```

Case Two: Deleting Object Tags (Versioning Enabled)

Request

The following request deletes existing tags from the object exampleobject.txt in the bucket

examplebucket-1250000000 , specifying the version ID of the object as MTg0NDUxNTgwODg2MTEzMTQyOTI . After parsing this request, COS deletes all tags in the object of this version.

```
DELETE /exampleobject.txt?tagging&VersionId=MTg0NDUxNTgwODg2MTEzMTQyOTI  HTTP/1.1
User-Agent: curl/7.29.0
Accept: */*
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Authorization: Auth String
Content-Length: 0
Content-Type: application/xml
```

Response

```
HTTP/1.1 204 No Content
Content-Type: application/xml
Connection: close
Date: Fri, 19 Jan 2020 11:45:22 GMT
```

Multipart Upload
Initializing Multipart Upload

Description

The Initiate Multipart Upload API request initializes shard upload, and upon successful execution, it returns an UploadId for subsequent Upload Part requests.

Request

Request Example

```
POST /<ObjectKey>?uploads HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (For details, please refer to the Request Signature document).

Request Header

Common Headers

The implementation of this request operation uses common request headers. For details on common request headers, please refer to the Common Request Headers document.

Non-common Headers

Recommended headers:
This request is implemented by using the following recommended request headers.

Name	Description	Type	Required
Cache-Control	Cache policy defined in RFC 2616. It is saved as object metadata.	String	No
Content-Disposition	File name defined in RFC 2616. It is saved as object metadata.	String	No

Name	Description	Type	Required
Content-Encoding	Encoding format defined in RFC 2616. It is saved as object metadata.	String	No
Content-Type	Content type (MIME) defined in RFC 2616. It is saved as object metadata.	String	No
Expires	The file date and time defined in RFC 2616 will be saved as Object metadata.	String	No
x-cos-meta-*	Including user-defined header suffixes and user-defined header information will be returned as Object metadata, with a size limit of 2 KB. Note: User-defined header information supports underscores, but user-defined header suffixes do not support underscores.	String	No

Permission related header

> Note:
>
> For more information on ACL requests, please refer to the ACL Overview document.

Name	Description	Type	Required
x-cos-acl	Defines the ACL property of the Object, with valid values: private, public-read, default; Default value: default (inherits Bucket permissions). Note: The current access policy entry limit is 1000 entries. If you do not need to control Object ACLs, please fill in default or leave this item unset, which will default to inheriting Bucket permissions.	String	No
x-cos-grant-read	Grants the grantee read permission, format: x-cos-grant-read: id="[OwnerUin]"	String	No
x-cos-grant-full-control	Grants the grantee full control permission, format: x-cos-grant-full-control: id="[OwnerUin]"	String	No

Request Body

The request operation's request body is empty.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Common Response Headers.

Response Body

The response body is returned as application/xml. An example including the complete node data is shown below:

```
<InitiateMultipartUploadResult>
  <Bucket>examplebucket-1250000000</Bucket>
  <Key>exampleobject</Key>
  <UploadId>1484727270323ddb949d528c629235314a9ead80f0ba5d993a3d76b460e6a9cceb9633b08e</UploadId>
</InitiateMultipartUploadResult>
```

The data are as follows:

Node Name (Keyword)	Parent Node	Description	Type
InitiateMultipartUploadResult	No	Description of all returned information	Container

Content of the Container node InitiateMultipartUploadResult:

Node Name (Keyword)	Parent Node	Description	Type
Bucket	InitiateMultipartUploadResult	Target Bucket for multipart upload, consisting of a user-defined string and a system-generated APPID digit string connected by a hyphen, for example: examplebucket-1250000000	Container
Key	InitiateMultipartUploadResult	Name of an object.	Container
UploadId	InitiateMultipartUploadResult	ID to use in subsequent uploads.	Container

Practical Case

Request

```
POST /exampleobject?uploads HTTP/1.1
```

```
Host: <BucketName-APPID>.<Endpoint>
Date: Fri, 10 Mar 2016 09:45:46 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84727259;32557623259&q-key-time=1484727259;32557623259&q-header-list=host&q-url-param-list
=uploads&q-signature=b5f46c47379aeaee74be7578380b193c01b28045
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 230
Connection: keep-alive
Date: Fri, 10 Mar 2016 09:45:46 GMT
x-cos-request-id: NTg3ZjIzZTZfOWIxZjRlXzZmMzhfMWRj

<InitiateMultipartUploadResult>
   <Bucket>examplebucket-1250000000</Bucket>
   <Key>exampleobject</Key>
   <UploadId>1484727270323ddb949d528c629235314a9ead80f0ba5d993a3d76b460e6a9cceb9633b0
8e</UploadId>
</InitiateMultipartUploadResult>
```

Uploading Parts

Description

The Upload Part API is used to upload an object to COS in parts. Up to 10,000 parts are supported, each part is 1 MB to 5 GB, and the last part can be less than 1 MB.

Notes

1. Multipart upload requires initialization first via the Initiate Multipart Upload interface, which returns an uploadId to uniquely identify the upload.
2. Each Upload Part request must include the partNumber and uploadId, where partNumber represents the part number and supports out-of-order uploading.
3. When the uploaded uploadId and partNumber are the same, the later part will overwrite the earlier part. When uploadId does not exist, a 404 error, NoSuchUpload, is returned.

Request

Syntax example:

```
PUT /ObjectName?partNumber=PartNumber&uploadId=UploadId HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Content-Length: Size
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (See Request Signature for details.)

Request Line

```
PUT /ObjectName?partNumber=PartNumber&uploadId=UploadId HTTP/1.1
```

This API accepts PUT requests.

Request Parameters

An example request line containing all request parameters:

```
PUT /ObjectName?partNumber=PartNumber&uploadId=UploadId HTTP/1.1
```

The specific contents are as follows:

Parameter Name	Description	Type	Required
partNumber	Identifies the number of the parts being uploaded in this multipart upload.	String	Yes
uploadId	The ID that identifies this multipart upload. When you use the Initiate Multipart Upload API to initiate multipart upload, you will get an uploadId, which not only uniquely identifies the multipart data, but also identifies the relative position of the multipart data in the entire file.	String	Yes

Request Header

Common Headers

This request uses common request headers. For details, see Common Request Headers.

Non-common Headers

Required Headers

This request operation requires the mandatory use of request headers, with specific contents as follows:

Name	Description	Type	Required
Content-Length	HTTP response length in bytes as defined in RFC 2616.	String	Yes

Recommended Headers

This request operation recommends the use of recommended headers in the request, with specific contents as follows:

Name	Description	Type	Required
Expect	HTTP response length in bytes as defined in RFC 2616.	String	No
Content-MD5	Base64-encoded 128-bit content MD5 checksum as defined in RFC 1864. This header is used to verify whether the file content has changed.	String	No

Request Body

The request operation's request body is empty.

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Public Response Header.

Response Body

The response body of this response is empty.

Practical Case

Request

```
PUT /ObjectName?partNumber=1&uploadId=1484727270323ddb949d528c629235314a9ead80f0ba5d
993a3d76b460e6a9cceb9633b08e HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Wed，18 Jan 2017 16:17:03 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84727403;32557623403&q-key-time=1484727403;32557623403&q-header-list=host&q-url-param-list
=partNumber;uploadId&q-signature=bfc54518ca8fc31b3ea287f1ed2a0dd8c8e88a1d
Content-Length: 10485760

[Object]
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Connection: keep-alive
Date: Wed，18 Jan 2017 16:17:03 GMT
Etag: "e1e5b4965bc7d30880ed6d226f78a5390f1c09fc"
x-cos-request-id: NTg3ZjI0NzlfOWIxZjRlXzZmNGJfMWYy
```

Copying Parts

Description

Upload Part - Copy request copies a segment of an object from the source path to the target path. Specify the source object with x-cos-copy-source, and the byte range with x-cos-copy-source-range (allowable chunk size is 5MB - 5GB).

> Note:
>
> - If the target object and source object are not in the same region, and the target object's chunk will exceed 5GB, then it's necessary to use the chunked upload or chunked copy interface to copy the object.
> - To use upload part object, you must first initialize a chunked upload. In the response of initializing a chunked upload, a unique descriptor (upload ID) will be returned. You need to carry this ID in the chunked upload request.

Version

When bucket version control is enabled, x-cos-copy-source identifies the current version of the object being copied. If the current version is a delete marker, and x-cos-copy-source does not specify a version, then COS will consider the object deleted and return a 404 error. If you specify a versionId in x-cos-copy-source and the versionId is a delete marker, then COS will return an HTTP 400 error, because delete markers are not allowed as versions for x-cos-copy-source.

Request

Request Example

```
PUT /examplebucket?partNumber=PartNumber&uploadId=UploadId  HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
x-cos-copy-source: <BucketName-APPID>.<Endpoint>/filepath
x-cos-copy-source-range: bytes=first-last
x-cos-copy-source-if-match: etag
x-cos-copy-source-if-none-match : etag
x-cos-copy-source-if-unmodified-since: time_stamp
x-cos-copy-source-if-modified-since: time_stamp
```

Request Header

Common Headers

This request operation is implemented using a public request header.

Non-common Headers

Required Headers

This request operation uses the following required headers:

Name	Description	Type	Required
x-cos-copy-source	The source object URL path, which can specify historical versions with the versionid subresource.	String	Yes

Recommended headers

This request operation uses the following recommended headers:

Name	Description	Type	Required
x-cos-copy-source-range	The byte range of the source object, the range value must use the bytes=first-last format, where both first and last are offsets starting from 0. For example, bytes=0-9 means you wish to copy the first 10 bytes of data from the source object. If not specified, it means copying the entire object.	String	No
x-cos-copy-source-If-Modified-Since	If the object is modified after the specified time, the operation will be performed; otherwise, 412 will be returned. This parameter can be used together with x-cos-copy-source-If-None-Match, returns conflict when used in conjunction with other conditions.	String	No
x-cos-copy-source-If-Unmodified-Since	If the object is not modified after the specified time, the operation will be performed; otherwise, 412 will be returned. This parameter can be used with x-cos-copy-source-If-Match, returns conflict when used in conjunction with other conditions.	String	No
x-cos-copy-source-If-Match	If the ETag of the object is the same as the specified one, the operation will be performed; otherwise, 412 will be returned. This parameter can be used with x-cos-copy-source-If-Unmodified-Since, returns conflict when used in conjunction with other conditions.	String	No

Name	Description	Type	Required
x-cos-copy-source-If-None-Match	If the ETag of the object is different from the specified one, the operation will be performed; otherwise, 412 will be returned. This parameter can be used with x-cos-copy-source-If-Modified-Since, returns conflict when used in conjunction with other conditions.	String	No

Request Parameters

Name	Description	Type	Required
partNumber	Part number	String	Yes
uploadId	To upload an object in parts, you must first initialize the multipart upload. The response of the multipart upload initialization will carry a unique descriptor (uploadId), which needs to be carried in the multipart upload request.	String	Yes

Request Body

The request body of this request is empty.

Response

Response Headers

Common Response Headers

This response contains common response headers.

Special Response Headers

Name	Description	Type
x-cos-copy-source-version-id	If version control has already been enabled on the source bucket, then the version of the source object will be replicated.	String
x-cos-server-side-encryption	If the object is stored with COS-managed server-side encryption, the response will contain this header and the encryption algorithm used (AES256).	String

Response Body

The response body returns application/xml data. The following contains all the nodes:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CopyPartResult>
  <ETag>"ba82b57cfdfda8bd17ad4e5879ebb4fe"</ETag>
  <LastModified>2017-09-04T04:45:45</LastModified>
</CopyPartResult>
```

The data are as follows:

Name	Description	Type
CopyPartResult	Returns the copy result information	String
ETag	Returns the MD5 algorithm checksum of the object. The value of ETag can be used to check if the content of the Object has changed.	String
LastModified	Last modification time of the object, in GMT format.	String

Practical Case

Request

```
PUT /exampleobject?partNumber=1&uploadId=1505706248ca8373f8a5cd52cb129f4bcf85e11dc8833df34f4f5bcc456c99c42cd1ffa2f9 HTTP/1.1
User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.13.1.0 zlib/1.2.3 libidn/1.18 libssh2/1.2.2
Accept: */*
x-cos-copy-source:<BucketName-APPID>.<Endpoint>/exampleobject1
x-cos-copy-source-range: bytes=10-100
Host: <BucketName-APPID>.<Endpoint>
Authorization:q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=1507530223;1508530223&q-key-time=1507530223;1508530223&q-header-list=&q-url-param-list=&q-signature=d02640c0821c49293e5c289fa07290e6b2f05cb2
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 133
Connection: keep-alive
Date: Mon, 04 Sep 2017 04:45:45 GMT
x-cos-request-id: NTlkYjFjYWJfMjQ4OGY3MGFfNGIzZV9k

<CopyPartResult>
  <ETag>"ba82b57cfdfda8bd17ad4e5879ebb4fe"</ETag>
```

```
  <LastModified>2017-09-04T04:45:45</LastModified>
</CopyPartResult>
```

Completing Multipart Upload

Description

The Complete Multipart Upload API is used to complete the entire multipart upload process. After all parts are uploaded by using the Upload Parts API, this API should be called to complete the multipart upload process of the entire file. When using this API, you should provide the PartNumber and ETag of each part in the request body to verify the accuracy of each part.
Since parts need to be merged after they are uploaded and merging takes a few minutes, COS immediately returns a 200 status code when merging begins. COS periodically returns space information to keep the connection alive until the merge is complete. COS returns the merged parts in the body.

- When the uploaded part is less than 1MB, calling this API will return a 400 EntityTooSmall.
- When the uploaded part numbers are not consecutive, calling this API will return 400 InvalidPart.
- When the part information in the request body is not arranged in ascending order by sequence number, calling this API will return a 400 InvalidPartOrder.
- When the UploadId does not exist, calling this API will return a 404 NoSuchUpload.

> Note:
>
> It is recommended to complete multipart uploads promptly or abort multipart uploads because being uploaded but unterminated parts will occupy storage space and incur storage fees.

Request

Request Example

```
POST /<ObjectKey>?uploadId=UploadId HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Content-length: Size
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (For details, please refer to section Request Signature).

Request Header

Common Headers

This request uses common request headers. For details, see Common Request Headers.

Non-common Headers

This request has no special request header information.

Request Body

The specific node content of the API request body is:

```
<CompleteMultipartUpload>
  <Part>
    <PartNumber>1</PartNumber>
    <ETag>"fc392a65890e447ff4e2d256489a9773"</ETag>
  </Part>
  <Part>
    <PartNumber>2</PartNumber>
    <ETag>"fc392a65890e447ff4e2d256489a9773"</ETag>
  </Part>
    ...
</CompleteMultipartUpload>
```

The data are as follows:

Node Name (Keyword)	Parent Node	Description	Type	Required
CompleteMultipartUpload	No	Information describing all aspects of this multipart upload	Container	Yes

Content of the Container node CompleteMultipartUpload:

Node Name (Keyword)	Parent Node	Description	Type	Required
Part	CompleteMultipartUpload	Information describing each part uploaded in this multipart upload	Container	Yes

Container node Part content:

Node Name (Keyword)	Parent Node	Description	Type	Required
PartNumber	CompleteMultipartUpload.Part	Part numbers	Integer	Yes
ETag	CompleteMultipartUpload.Part	MD5 checksum value of each block file	String	Yes

Response

Response Headers

Common Response Headers

This response uses common response headers. For details, see Public Response Header.

Response Body

The response body returns application/xml data. The following contains all the nodes:

```
<CompleteMultipartUploadResult>
  <Location>examplebucket-1250000000.cos.city.yfm4.fsphere.cn/ObjectName</Location>
  <Bucket>examplebucket-1250000000</Bucket>
  <Key>examplebucket</Key>
  <ETag>"3a0f1fd698c235af9cf098cb74aa25bc"</ETag>
</CompleteMultipartUploadResult>
```

The data are as follows:

Node Name (Keyword)	Parent Node	Description	Type
CompleteMultipartUploadResult	No	Description of all returned information	Container

Content of Container node CompleteMultipartUploadResult:

Node Name (Keyword)	Parent Node	Description	Type
Location	CompleteMultipartUploadResult	The public access domain name for creating the object.	URL
Bucket	CompleteMultipartUploadResult	The destination bucket for the multipart upload, consisting of a user-defined string concatenated with the system-generated appid numeric string, separated by a hyphen, such as: examplebucket-1250000000.	String
Key	CompleteMultipartUploadResult	Object name.	String
ETag	CompleteMultipartUploadResult	The unique tag value of the merged object, which is not the MD5 checksum of	String

Node Name (Keyword)	Parent Node	Description	Type
		the object content and is only used to check the uniqueness of the object.	

Practical Case

Request

```
POST /exampleobject?uploadId=1484728886e63106e87d8207536ae8521c89c42a436fe23bb58854a7b
b5e87b7d77d4ddc48 HTTP/1.1
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Date: Wed,18 Jan 2017 16:17:03 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84729794;32557625794&q-key-time=1484729794;32557625794&q-header-list=host&q-url-param-list
=uploadId&q-signature=23627c8fddb3823cce4257b33c663fd83f9f820d
Content-Length: 138

<CompleteMultipartUpload>
  <Part>
    <PartNumber>1</PartNumber>
    <ETag>"fc392a65890e447ff4e2d256489a9773"</ETag>
  </Part>
  <Part>
    <PartNumber>2</PartNumber>
    <ETag>"fc392a65890e447ff4e2d256489a9774"</ETag>
  </Part>
</CompleteMultipartUpload>
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 277
Connection: keep-alive
Date: Wed,18 Jan 2017 16:17:03 GMT
x-cos-request-id: NTg3ZjJlMjVfNDYyMDRlXzM0YzRfMjc1

<CompleteMultipartUploadResult>
    <Location>examplebucket-1250000000.cos.city.yfm4.fsphere.cn/ObjectName</Location>
    <Bucket>examplebucket-1250000000</Bucket>
    <Key>examplebucket</Key>
```

```
    <ETag>"3a0f1fd698c235af9cf098cb74aa25bc"</ETag>
</CompleteMultipartUploadResult>
```

Terminating Multipart Upload

Description

Abort Multipart Upload is used to abort multipart upload and delete the uploaded parts. When you call Abort Multipart Upload, if there is a request that is using this Upload Parts to upload parts, an error is returned. When the UploadId does not exist, a 404 NoSuchUpload error will be returned.

> Note:
>
> It is recommended to complete multipart uploads promptly or abort multipart uploads because being uploaded but unterminated parts will occupy storage space and incur storage fees.

Request

Request Example

```
DELETE /<ObjectKey>?uploadId=UploadId HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (For details, please refer to the Request Signature documentation).

Request Parameters

The specific contents are as follows:

Parameter Name	Description	Type	Required
uploadId	The ID that identifies this multipart upload. When you use the Initiate Multipart Upload API to initialize multipart upload, you will get an uploadId, which not only uniquely identifies the multipart data, but also identifies the relative position of the multipart data in the entire file.	String	Yes

Request Header

Common Headers

The implementation of this request utilizes common request headers. For details on common request headers, please refer to the Common Request Headers documentation.

Non-common Headers

This request has no special request header information.

Request Body

The request body for this request is empty.

Response

Response Headers

Common Response Headers

This response utilizes common response headers. For details on common response headers, please refer to the Common Response Headers documentation.

Special Response Headers

There are no special response headers for this request operation.

Response Body

The response body of this request is empty.

Practical Case

Request

```
DELETE /exampleobject?uploadId=1484727270323ddb949d528c629235314a9ead80f0ba5d993a3d76b460e6a9cceb9633b08e HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: Tue, 26 Oct 2013 21:22:00 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=1484728626;32557624626&q-key-time=1484728626;32557624626&q-header-list=host&q-url-param-list=uploadId&q-signature=2d3036b57cade4a257b48a3a5dc922779a562b18
```

Response

```
HTTP/1.1 204 OK
Content-Type: application/xml
Content-Length: 0
```

Connection: keep-alive
Date: Tue, 26 Oct 2013 21:22:00 GMT
x-cos-request-id: NTg3ZjI5MzlfOTgxZjRlXzZhYjNfMjBh

Querying Multipart Upload

Description

List Multipart Uploads is used to query the parts being uploaded. A maximum of 1,000 parts being uploaded can be listed in a single request at a time.

> Note:
>
> Read permissions for the bucket are required for this request.

Request

Request Example

```
GET /?uploads HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (For details, please refer to the Request Signature document).

Request Header

Common Headers

This request uses common request headers. For details, see Common Request Headers.

Non-common Headers

This request has no special request header information.

Request Parameters

The specific contents are as follows:

Name	Description	Type	Required
delimiter	The delimiter is a symbol. Objects whose names contain the specified prefix and the first occurrence of the delimiter character are considered as a group of elements: common prefix. If there is no prefix, it starts from the beginning of the path.	String	No

Name	Description	Type	Required
encoding-type	Specifies the encoding format of the return value. Legal value: url	String	No
prefix	The returned Object key must be prefixed with Prefix. Note that if you use this parameter, the returned keys will contain this prefix.	String	No
max-uploads	Set the maximum number of multiparts returned. The legal value is from 1 to 1000. The default value is 1000.	String	No
key-marker	Used together with upload-id-marker When upload-id-marker is not specified, entries with ObjectNames alphabetically greater than key-marker will be listed. When upload-id-marker is specified, entries with an ObjectName alphabetically greater than key-marker are listed, and entries with an ObjectName alphabetically equal to key-marker and an UploadID greater than upload-id-marker are listed.	String	No
upload-id-marker	Used with key-marker When key-marker is not specified, upload-id-marker will be ignored. When key-marker is specified, entries with an ObjectName greater than key-marker in alphabetical order are listed, and entries with an ObjectName equal to key-marker in alphabetical order and an UploadID greater than upload-id-marker are listed.	String	No

Request Body

The request body of this request is empty.

Response

Response Headers

Common Response Headers

This response contains common response headers. For details on common response headers, please refer to the Common Response Header document.

Special Response Headers

There are no special response headers for this response operation.

Response Body

The response body returns application/xml data. The following contains all the nodes:

```
<ListMultipartUploadsResult>
  <Bucket></Bucket>
  <Encoding-Type></Encoding-Type>
  <KeyMarker></KeyMarker>
  <UploadIdMarker></UploadIdMarker>
  <NextKeyMarker></NextKeyMarker>
  <NextUploadIdMarker></NextUploadIdMarker>
  <MaxUploads></MaxUploads>
  <IsTruncated></IsTruncated>
  <Prefix></Prefix>
  <Delimiter></Delimiter>
  <Upload>
    <Key></Key>
    <UploadID></UploadID>
    <StorageClass></StorageClass>
    <Initiator>
      <ID></ID>
    <DisplayName></DisplayName>
    </Initiator>
    <Owner>
      <ID></ID>
    <DisplayName></DisplayName>
    </Owner>
    <Initiated></Initiated>
  </Upload>
  <CommonPrefixes>
    <Prefix></Prefix>
  </CommonPrefixes>
</ListMultipartUploadsResult>
```

The data are as follows:

Node Name (Keyword)	Parent Node	Description	Type
ListMultipartUploadsResult	No	Used to describe all the information about the multipart upload	Container

Contents of the Container node ListMultipartUploadsResult:

Node Name (Keyword)	Parent Node	Description	Type
Bucket	ListMultipartUploadsResult	Target Bucket for multipart	String

Node Name (Keyword)	Parent Node	Description	Type
		upload, consisting of a user-defined string and a system-generated APPID digit string connected by a hyphen, for example: examplebucket-1250000000	
Encoding-Type	ListMultipartUploadsResult	Specifies the encoding format of the return value. Legal value: url	String
KeyMarker	ListMultipartUploadsResult	List entries starting from this key value	String
UploadIdMarker	ListMultipartUploadsResult	List entries starting with this UploadId value	String
NextKeyMarker	ListMultipartUploadsResult	If the returned entry is truncated, the returned NextKeyMarker is the starting point of the next entry.	String
NextUploadIdMarker	ListMultipartUploadsResult	If the returned entry is truncated, the returned UploadId is the starting point of the next entry.	String
MaxUploads	ListMultipartUploadsResult	Set the maximum number of multiparts returned. The legal value is from 0 to 1000.	String
IsTruncated	ListMultipartUploadsResult	Returns whether the entry is truncated. Boolean value is TRUE or FALSE.	Boolean
Prefix	ListMultipartUploadsResult	A prefix that the returned object keys must start with. Note that if you use this parameter, the returned keys will contain this prefix.	String
Delimiter	ListMultipartUploadsResult	The delimiter is a symbol. The objects whose names contain the specified prefix and the first occurrence of the delimiter character are regarded as a group of elements: common prefix. If there is no prefix, it starts from	String

Node Name (Keyword)	Parent Node	Description	Type
		the beginning of the path.	
Upload	ListMultipartUploadsResult	Information about each upload	Container
CommonPrefixes	ListMultipartUploadsResult	Group the same paths from prefix to delimiter into one category and define it as Common Prefix	Container

Contents of the Container node upload:

Node Name (Keyword)	Parent Node	Description	Type
Key	ListMultipartUploadsResult.Upload	Name of an object.	String
UploadID	ListMultipartUploadsResult.Upload	ID of the multipart upload	String
StorageClass	ListMultipartUploadsResult.Upload	Used to indicate the storage level of parts, enumeration value: STANDARD, STANDARD_IA, and ARCHIVE	String
Initiator	ListMultipartUploadsResult.Upload	Used to indicate the information of the initiator of this upload	Container
Owner	ListMultipartUploadsResult.Upload	Used to indicate the information of the owners of these parts	Container
Initiated	ListMultipartUploadsResult.Upload	Start time of multipart upload.	Date

Contents of the Container node Initiator:

Node Name (Keyword)	Parent Node	Description	Type
ID	ListMultipartUploadsResult.Upload.Initiator	User's unique CAM ID	String
DisplayName	ListMultipartUploadsResult.Upload.Initiator	Abbreviation for User Identity Number (UIN)	String

Content of Owner:

Node Name (Keyword)	Parent Node	Description	Type
ID	ListMultipartUploadsResult.Upload.Owner	User's unique CAM ID	String
DisplayName	ListMultipartUploadsResult.Upload.Owner	Abbreviation for User Identity Number (UIN)	String

Contents of the Container node CommonPrefixes:

Node Name (Keyword)	Parent Node	Description	Type
Prefix	ListMultipartUploadsResult.CommonPrefixes	Display specific CommonPrefixes	String

Error Analysis

The following describes some special but common errors that may occur with this request:

Error Code	HTTP Status Code	Description
InvalidArgument	400 Bad Request	max-uploads must be an integer between 0 and 1000, otherwise InvalidArgument is returned. encoding-type can only be set to url; otherwise, InvalidArgument will be returned.

For more information on COS error codes or the complete list of product errors, please refer to the Error Codes document.

Practical Case

Request

```
GET /?uploads HTTP/1.1
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Date: Wed, 18 Jan 2015 21:32:00 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84727508;32557623508&q-key-time=1484727508;32557623508&q-header-list=host&q-url-param-list
=uploads&q-signature=5bd4759a7309f7da9a0550c224d8c61589c9dbbf
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 1203
Date: Wed, 18 Jan 2015 21:32:00 GMT
x-cos-request-id: NTg3ZjI0ZGRfNDQyMDRlXzNhZmRfMjRl

<ListMultipartUploadsResult>
   <Bucket>examplebucket-1250000000</Bucket>
   <Encoding-Type/>
   <KeyMarker/>
   <UploadIdMarker/>
   <MaxUploads>1000</MaxUploads>
   <Prefix/>
   <Delimiter>/</Delimiter>
   <IsTruncated>false</IsTruncated>
   <Upload>
      <Key>Object</Key>
      <UploadID>1484726657932bcb5b17f7a98a8cad9fc36a340ff204c79bd2f51e7dddf0b6d1da62205
20c</UploadID>
      <Initiator>
        <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
      <DisplayName>100000000001</DisplayName>
      </Initiator>
      <Owner>
        <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
      <DisplayName>100000000001</DisplayName>
      </Owner>
      <StorageClass>Standard</StorageClass>
      <Initiated>Wed Jan 18 16:04:17 2017</Initiated>
   </Upload>
   <Upload>
      <Key>Object</Key>
      <UploadID>1484727158f2b8034e5407d18cbf28e84f754b791ecab607d25a2e52de9fee641e5f607
07c</UploadID>
      <Initiator>
        <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
      <DisplayName>100000000001</DisplayName>
      </Initiator>
      <Owner>
        <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
      <DisplayName>100000000001</DisplayName>
      </Owner>
      <StorageClass>Standard</StorageClass>
      <Initiated>Wed Jan 18 16:12:38 2017</Initiated>
   </Upload>
   <Upload>
```

```
        <Key>exampleobject</Key>
        <UploadID>1484727270323ddb949d528c629235314a9ead80f0ba5d993a3d76b460e6a9cceb9633
b08e</UploadID>
        <Initiator>
          <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
        <DisplayName>100000000001</DisplayName>
        </Initiator>
        <Owner>
          <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
        <DisplayName>100000000001</DisplayName>
        </Owner>
        <StorageClass>Standard</StorageClass>
        <Initiated>Wed Jan 18 16:14:30 2017</Initiated>
    </Upload>
</ListMultipartUploadsResult>
```

Querying Uploaded Parts

Description

List Parts is used to query the uploaded parts in a specific multipart upload, that is, to list all successfully uploaded parts belonging to the specified UploadId.

Request

Request Example

```
GET /<ObjectKey>?uploadId=UploadId HTTP/1.1
Host: <BucketName-APPID>.<Endpoint>
Date: GMT Date
Authorization: Auth String
```

> Note:
>
> Authorization: Auth String (For details, see Request Signature.)

Request Header

Common Headers

This request uses common request headers. For details, see Common Request Headers.

Non-common Headers

This request has no special request header information.

Request Parameters

Name	Type	Required	Description
UploadId	string	Yes	The ID that identifies this multipart upload. When you use the Initiate Multipart Upload API to initialize multipart upload, you will get an uploadId, which not only uniquely identifies the multipart data, but also identifies the relative position of the multipart data in the entire file.
encoding-type	string	No	Encoding method of the return value.

Name	Type	Required	Description
max-parts	string	No	Maximum number of entries returned in a single response. The default value is 1,000.
part-number-marker	string	No	By default, entries are listed in UTF-8 binary order, starting from the marker.

Request Body

The request does not have a request body.

Response

Response Headers

Common Response Headers

This response contains common response headers. For details, see Common Response Headers.

Special Response Headers

There are no special response headers for this response operation.

Response Body

The query is successful and returns application/xml data, including completed shard information.

```
<?xml version="1.0" encoding="UTF-8" ?>
<ListPartsResult>
  <Bucket>examplebucket-1250000000</Bucket>
  <Encoding-type/>
  <Key>exampleobject</Key>
  <UploadId>14846420620b1f381e5d7b057692e131dd8d72dfa28f2633cfbbe4d0a9e8bd0719933545
b0</UploadId>
  <Initiator>
    <ID>1250000000</ID>
    <DisplyName>1250000000</DisplyName>
  </Initiator>
  <Owner>
    <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
    <DisplyName>100000000001</DisplyName>
  </Owner>
  <PartNumberMarker>0</PartNumberMarker>
  <Part>
```

```
        <PartNumber>1</PartNumber>
        <LastModified>Tue Jan 17 16:43:37 2017</LastModified>
        <ETag>"a1f8e5e4d63ac6970a0062a6277e191fe09a1382"</ETag>
        <Size>5242880</Size>
    </Part>
    <NextPartNumberMarker>1</NextPartNumberMarker>
    <StorageClass>STANDARD</StorageClass>
    <MaxParts>1</MaxParts>
    <IsTruncated>true</IsTruncated>
</ListPartsResult>
```

The data are described as follows:

Node Name (Keyword)	Parent Node	Description	Type
ListPartsResult	No	Save all information from the List Parts request result.	Container

Content of the Container node ListPartsResult:

Node Name (Keyword)	Parent Node	Description	Type
Bucket	ListPartsResult	The target Bucket for multipart upload, the name of the bucket, consisting of a user-defined string and a system-generated APPID number connected by hyphens, for example: examplebucket-1250000000.	string
Encoding-Type	ListPartsResult	Encoding Format	string
Key	ListPartsResult	The name of the Object.	string
UploadId	ListPartsResult	The ID that identifies this multipart upload	string
Initiator	ListPartsResult	Used to indicate the information of the owners of these parts	Container
Owner	ListPartsResult	Used to indicate the information of the owners of these parts	Container
StorageClass	ListPartsResult	The storage class of these parts, with enumerated values: STANDARD, STANDARD_IA, ARCHIVE.	string
PartNumberMarker	ListPartsResult	By default, entries are listed in UTF-8 binary order, starting from the marker.	string

Node Name (Keyword)	Parent Node	Description	Type
NextPartNumberMarker	ListPartsResult	If the returned entries are truncated, NextMarker is the starting point of the next entry.	string
MaxParts	ListPartsResult	The maximum number of entries to return at a time	string
IsTruncated	ListPartsResult	Indicates whether the returned list is truncated. Valid Boolean value: true, false.	boolean
Part	ListPartsResult	Metadata Information	Container

Contents of the Container node Initiator:

Node Name (Keyword)	Parent Node	Description	Type
ID	ListPartsResult.Initiator	Creator's unique identifier	string
DisplayName	ListPartsResult.Initiator	Description of the creator's username	string

Content of Owner:

Node Name (Keyword)	Parent Node	Description	Type
ID	ListPartsResult.Owner	Creator's unique identifier	string
DisplayName	ListPartsResult.Owner	Description of the creator's username	string

Container node Part content:

Node Name (Keyword)	Parent Node	Description	Type
PartNumber	ListPartsResult.Part	Part number	string
LastModified	ListPartsResult.Part	Description of when the block was last modified	string
ETag	ListPartsResult.Part	Block's MD-5 algorithm checksum	string
Size	ListPartsResult.Part	Description of block size, in bytes	string

Practical Case

Request

```
GET /exampleobject?uploadId=14846420620b1f381e5d7b057692e131dd8d72dfa28f2633cfbbe4d0a9e
8bd0719933545b0&max-parts=1 HTTP/1.1
Host: examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Date: Wed,18 Jan 2017 16:17:03 GMT
Authorization:q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=14
84643123;1484646723&q-key-time=1484643123;1484646723&q-header-list=host&q-url-param-list=
max-parts;uploadid&q-signature=b8b4055724e64c9ad848190a2f7625fd3f9d3e87
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 661
Connection: keep-alive
Date: Wed,18 Jan 2017 16:17:03 GMT
x-cos-request-id: NTg3ZGRiMzhfMmM4OGY3XzdhY2NfYw==

<ListPartsResult>
   <Bucket>examplebucket-1250000000</Bucket>
   <Encoding-type/>
   <Key>exampleobject</Key>
   <UploadId>14846420620b1f381e5d7b057692e131dd8d72dfa28f2633cfbbe4d0a9e8bd0719933545
b0</UploadId>
   <Initiator>
      <ID>1250000000</ID>
      <DisplyName>1250000000</DisplyName>
   </Initiator>
   <Owner>
      <ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
      <DisplyName>100000000001</DisplyName>
   </Owner>
   <PartNumberMarker>0</PartNumberMarker>
   <Part>
      <PartNumber>1</PartNumber>
      <LastModified>Tue Jan 17 16:43:37 2017</LastModified>
      <ETag>"a1f8e5e4d63ac6970a0062a6277e191fe09a1382"</ETag>
      <Size>5242880</Size>
   </Part>
   <NextPartNumberMarker>1</NextPartNumberMarker>
   <StorageClass>STANDARD</StorageClass>
   <MaxParts>1</MaxParts>
   <IsTruncated>true</IsTruncated>
</ListPartsResult>
```

Data Processing APIs
Image Processing
Basic Image Processing
Adjusting Brightness

Feature Overview

Cloud Infinite (CI) adjusts the image brightness through the imageMogr2 API.

This feature supports the following processing methods.

- Processing upon download

Restrictions

- Supported formats: JPG, BMP, GIF, PNG, and WebP images can be processed, and HEIF images can be decoded and processed.
- Size limit: The size of the input image cannot be larger than 32 MB, its width and height cannot exceed 30,000 pixels, and the total number of pixels cannot exceed 250 million. The width and height of the output image cannot exceed 9,999 pixels. For an animated image, the product of width x height x FPS cannot exceed 250 million.
- FPS (for animated images): For GIFs, the FPS cannot exceed 300.

API Example

Processing upon Download

```
GET /<ObjectKey>?imageMogr2/bright/<value> HTTP/1.1
Host: <BucketName>.<Endpoint>
Date: <GMT Date>
Authorization: <Auth String>
```

> Note:
>
> Authorization: Auth String (For details, please refer to section Request Signature).

Processing Parameter Description

Operation name: bright

Parameter	Description
ObjectKey	File name of the object, for example, folder/sample.jpg.
/bright/<value>	Image brightness adjustment feature, where the value is the brightness parameter value. Value range: integers in [–100, 100]. - Value < 0: Decreases the image brightness. - Value = 0: Does not adjust the image brightness. - Value > 0: Increases the image brightness.
/ignore-error/1	If this parameter is carried and the image fails to be processed because the image is too large or a parameter value exceeds the limit, the input image is returned and no error is reported.

Practical Case

> Note:
>
> This type of processing does not save the output image to the bucket.

Example 1: Adjusting Brightness

This example increases the brightness of an image by 70:

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/bright/70
```

The following figure shows the final effect.

Notes

To prevent unauthorized users from accessing or downloading the input image by using a URL that does not carry any processing parameter, you can add the processing parameters to the request signature. The following is a simple example. For a detailed calculation method, see Request Signature.

http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-Signature=<signature-value>&watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWl5aW4uanBn/gravity/southeast

Removing Metadata

Feature Overview

CI removes the image metadata, including EXIF information through the imageMogr2 API.

This feature supports the following processing methods.

- Processing upon download

Restrictions

- Supported formats: JPG, BMP, GIF, PNG, and WebP images can be processed, and HEIF images can be decoded and processed.
- Size limit: The size of the input image cannot be larger than 32 MB, its width and height cannot exceed 30,000 pixels, and the total number of pixels cannot exceed 250 million. The width and height of the output image cannot exceed 9,999 pixels. For an animated image, the product of width x height x FPS cannot exceed 250 million.
- FPS (for animated images): For GIFs, the FPS cannot exceed 300.

API Example

1. Processing upon Download

```
GET /<ObjectKey>?imageMogr2/strip HTTP/1.1
Host: <BucketName>.<Endpoint>
Date: <GMT Date>
Authorization: <Auth String>
```

> Note:
>
> Authorization: Auth String (For details, please refer to section Request Signature).

Processing Parameter Description

Operation name: strip

Parameter	Description

Parameter	Description
ObjectKey	File name of the object, for example, folder/sample.jpg.
/ignore-error/1	If this parameter is carried and the image fails to be processed because the image is too large or a parameter value exceeds the limit, the input image is returned and no error is reported.

Practical Case

> Note:
>
> This type of processing does not save the output image to the bucket.

Example 1: Removing Metadata

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/strip
```

Notes

To prevent unauthorized users from accessing or downloading the input image by using a URL that does not carry any processing parameter, you can add the processing parameters to the request signature. The following is a simple example. For a detailed calculation method, see Request Signature.

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-Signature=<signature-value>&watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWl5aW4uanBn/gravity/southeast
```

Image Watermark

Feature Overview

CI provides an image watermark processing feature through the watermark API. The watermark image should be already stored in Cloud Infinite (CI). The size of the input image cannot exceed 32 MB, its width and height cannot exceed 30,000 pixels, and the total number of pixels cannot exceed 250 million. The width and height of the output image cannot exceed 9,999 pixels. For animated images, the product of width x height x FPS cannot exceed 250 million pixels.

This feature supports the following processing methods.

- Processing upon download

> Note:
>
> - Only one image watermark can be added to one image.
> - An animated image cannot be used as a watermark.

API Example

1. Processing upon Download

```
GET /<ObjectKey>?watermark/1/image/<encodedURL>
                /gravity/<gravity>
                /dx/<dx>
                /dy/<dy>
                /blogo/<type> HTTP/1.1
Host: <BucketName>.<Endpoint>
Date: <GMT Date>
Authorization: <Auth String>
```

> Note:
>
> - Spaces and line breaks in the preceding figure can be ignored.
> - Authorization: Auth String (For details, please refer to section Request Signature).

Processing Parameter Description

Operation name: watermark. The number 1 indicates that the watermark type is an image.

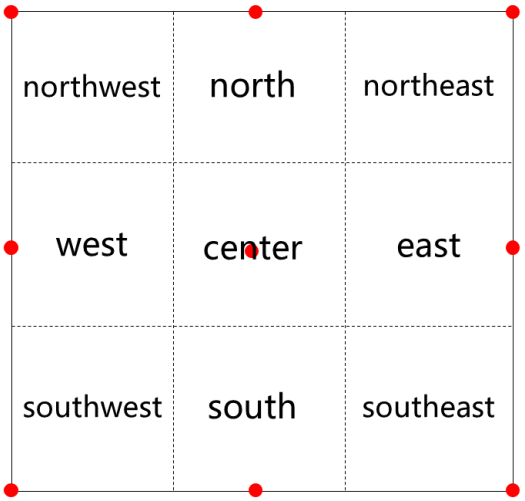
Parameter	Description
ObjectKey	File name of the object, for example, folder/sample.jpg.
/image/	URL-safe Base64-encoded URL of the image watermark. For example, if the watermark image is http://examplebucket-1250000000.cos.city.yfm4.fsphere.com/shuiyin_2.png , the encoded string is aHR0cDovL2V4YW1wbGVidWNrZXQtMTI1MDAwMDAwMC5jb3MuYXAtc2hhbmdoYWkubXlxY2xvdWQuY29tL3NodWI5aW5fMi5wbmc
/gravity/	Position of the image watermark in a Sudoku grid layout (See Sudoku Orientation Diagram). The default value is SouthEast.
/dx/	Horizontal (X-axis) margin, in pixels. The default value is 0.
/dy/	Vertical (Y-axis) margin, in pixels. The default value is 0.
/blogo/	Watermark image adaptation feature, which is suitable for scenarios where the watermark image size is too large (such as a watermark wall). There are two types: • When blogo is set to 1, the watermark image is scaled to a size similar to the input image before being added. • When blogo is set to 2, the watermark image is cropped to a size similar to the input image before being added.
/scatype/	The watermark image is scaled based on the size of the input image. - When scatype is set to 1, the watermark image is scaled based on the width of the input image. - When scatype is set to 2, the watermark image is scaled based on the height of the input image. • When scatype is set to 3, the watermark image is scaled based on the overall area of the input image.

Parameter	Description
/spcent/	It is used together with scatype. - When scatype is set to 1, the valid range of this parameter is [1,1000], in permille (‰). - When scatype is set to 2, the valid range of this parameter is [1,1000], in permille (‰). - When scatype is set to 3, the valid range of this parameter is [1,250], in permille (‰). Example: http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?watermark/1/image/xxxxxxx/scatype/3/spcent/250
/dissolve/	Transparency of the image watermark. The value range is 1–100. The default value is 90 (90% opacity).
/batch/	Watermark tiling feature, which allows the image watermark to be tiled over the entire image. When the value is 1, the watermark tiling feature is enabled.
/degree/	Rotation angle of the image watermark. It is valid only when batch is set to 1. The value range is 0 to 360, and the default value is 0.

> Note:
>
> A specified image watermark should meet the following three conditions:
>
> - The URL should be a COS domain name (CDN accelerated domain names cannot be used, such as examplebucket-1250000000.cos.city.yfm4.fsphere.cn/shuiyin_2.png), and the watermark image is accessible (if the watermark image read permission is private, a valid signature needs to be carried).
> - The URL should start with http:// , cannot omit the HTTP header, and cannot use an HTTPS header. For example, examplebucket-1250000000.cos.city.yfm4.fsphere.cn/shuiyin_2.png and https://examplebucket-1250000000.cos.city.yfm4.fsphere.cn/shuiyin_2.png are illegal watermark URLs.

Sudoku Orientation Diagram

The Sudoku orientation diagram provides reference positions for various operations on the image. The red dot represents the origin of each region (After selecting a region through the gravity parameter, displacement operations refer to the corresponding origin).



> Note:
>
> - When the gravity parameter is set to center, the dx and dy parameters are invalid.
> - When the gravity parameter is set to north or south, the dx parameter is invalid (the watermark is horizontally centered).
> - When the gravity parameter is set to west or east, the dy parameter is invalid (the watermark is vertically centered).

Practical Case

> Note:
>
> This type of processing does not save the output image to the bucket.

Example 1: Adding an Image Watermark

http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWl5aW4uanBn/gravity/southeast

The following figure shows the effect after an image watermark is added.

Example 2: Adding an Image Watermark with the Tile Mode and Opacity Specified

https://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWl5aW4uanBn/gravity/southeast/batch/1/degree/45/dissolve/40/

The following figure shows the effect of an added image watermark rotated by 45 degrees and tiled with 40% opacity.

Notes

To prevent unauthorized users from accessing or downloading the input image by using a URL that does not carry any processing parameter, you can add the processing parameters to the request signature. The following is a simple example. For a detailed calculation method, see Request Signature.

http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-Signature=<signature-value>&watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWI5aW4uanBn/gravity/southeast

Contrast

Feature Overview

Contrast is the measurement of different luminance levels between the brightest white and the darkest black in light and dark areas of an image, which indicates the degree of grayscale contrast in an image. CI adjusts the image contrast through the imageMogr2 API.

This feature supports the following processing methods.

- Processing upon download

Restrictions

- Supported formats: JPG, BMP, GIF, PNG, and WebP images can be processed, and HEIF images can be decoded and processed.
- Size limit: The size of the input image cannot be larger than 32 MB, its width and height cannot exceed 30,000 pixels, and the total number of pixels cannot exceed 250 million. The width and height of the output image cannot exceed 9,999 pixels. For an animated image, the product of width x height x FPS cannot exceed 250 million.
- FPS (for animated images): For GIFs, the FPS cannot exceed 300.

API Example

1. Processing upon Download

```
GET /<ObjectKey>?imageMogr2/contrast/<value> HTTP/1.1
Host: <BucketName>.<Endpoint>
Date: <GMT Date>
Authorization: <Auth String>
```

> Note:
>
> Authorization: Auth String (For details, please refer to section Request Signature).

Processing Parameter Description

Operation name: contrast.

Parameter	Description
ObjectKey	File name of the object, for example, folder/sample.jpg.
/contrast/<value>	Image contrast adjustment feature, where the value is the contrast parameter value. Value range: integers in [–100, 100]. - Value < 0: Decreases the image contrast. - Value = 0: Does not adjust the image contrast. - Value > 0: Increases the image contrast.
/ignore-error/1	If this parameter is carried and the image fails to be processed because the image is too large or a parameter value exceeds the limit, the input image is returned and no error is reported.

Practical Case

> Note:
>
> This type of processing does not save the output image to the bucket.

Example 1: Adjusting the Contrast

This example reduces the contrast of an image by 50:

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/contrast/-50
```

The following figure shows the final effect.

Notes

To prevent unauthorized users from accessing or downloading the input image by using a URL that does not carry any processing parameter, you can add the processing parameters to the request signature. The following is a simple example. For a detailed calculation method, see Request Signature.

http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-Signature=<signature-value>&watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWl5aW4uanBn/gravity/southeast

Quick Thumbnail Template

Feature Overview

CI provides common image processing templates through the imageView2 API. Developers can simply append the required parameters to the download URL to generate the specified thumbnail. The size of the input image cannot exceed 32 MB, its width and height cannot exceed 30,000 pixels, and the total number of pixels cannot exceed 250 million. The width and height of the output image cannot exceed 9,999 pixels. For animated images, the product of width x height x FPS cannot exceed 250 million pixels, and the GIF FPS is limited to 300.

This feature supports the following processing methods.

- Processing upon download

API Example

1. Processing upon Download

```
GET /<ObjectKey>?imageView2/<mode>/w/<Width>
                        /h/<Height>
                        /format/<Format>
                        /q/<Quality>
                        /rq/<Quality>
                        /lq/<Quality> HTTP/1.1
Host: <BucketName>.<Endpoint>
Date: <GMT Date>
Authorization: <Auth String>
```

> Note:
>
> - Quality transformation parameters apply only to images in jpg and webp formats.
> - Authorization: Auth String (For details, please refer to section Request Signature).

Processing Parameter Description

Parameter	Description
ObjectKey	File name of the object, for example, folder/sample.jpg.

Parameter	Description
/1/w/<Width>/h/<Height>	Limits the minimum width and height of the thumbnail. This operation proportionally scales the image until one side reaches the set minimum value. Then, the other side is centered and cropped to the set value. If only one side is specified, it implies a square of equal width and height. For example, if the input image size is 1000 x 500 and the parameter is set to ?imageView2/1/w/500/h/400, the image is first scaled proportionally to 800 x 400 and then cropped 150 from each side to produce a 500 x 400 image.
/2/w/<Width>/h/<Height>	Limits the maximum width and height of the thumbnail. This operation scales the image proportionally until both the width and height are less than the set maximum values. For example, if the input image size is 1000 x 500 and the parameter is set to ?imageView2/2/w/500/h/400, the image is scaled proportionally to 500 x 250. If only one side is specified, the other side is adapted automatically.
/3/w/<Width>/h/<Height>	Limits the minimum width and height of the thumbnail. This operation scales the image proportionally until both the width and height of the image are greater than the set minimum values. For example, if the input image size is 1000 x 500 and the parameter is set to ?imageView2/3/w/500/h/400, the image is scaled proportionally to 800 x 400. If only one side is specified, the other side is set to the same value.
/4/w/<LongEdge>/h/<ShortEdge>	Limits the minimum values of the long side and short side of the thumbnail to LongEdge and ShortEdge, respectively, for proportional compression. If only one side is specified, the other side has the same value.
/5/w/<LongEdge>/h/<ShortEdge>	Limits the maximum values of the long side and short side of the thumbnail to LongEdge and ShortEdge, respectively, for proportional compression and centered cropping. If only one side is specified, it indicates a square with equal width and height. After scaling, excess parts will be cropped.
/format/<Format>	Format of the target thumbnail, which can be JPG, BMP, GIF, PNG, or WebP. Default value: format of the input image
/q/<Quality>	Image quality. Value range: 0–100. The default value is the input image quality. The minimum value between the input image quality and the specified quality is taken. Add an exclamation mark (!) after (note it is an English character) to forcibly use the specified value.
/rq/<quality>	Relative quality of the image. Value range:0–100. The value is based on the input image quality. For example, if the input image quality is 80 and rquality is set to 80, the quality of the output image is 64 (80 x 80%).
/lq/<quality>	Minimum quality of the image. Value range: 0–100. It indicates the minimum quality value of the output image. For example, if the input image quality is 85 and lquality is set to 80, the quality

Parameter	Description
	of the output image is 85. For example, if the input image quality is 60 and lquality is set to 80, the quality of the output image is increased to 80.
/ignore-error/1	If this parameter is carried and the image fails to be processed because the image is too large or a parameter value exceeds the limit, the input image is returned and no error is reported.

Practical Case

> Note:
>
> This type of processing does not save the output image to the bucket.

Example 1: Using Template Style 1

The following example uses style 1, sets the minimum width and height of the thumbnail to 400 x 600, and sets the absolute quality to 85.

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageView2/1/w/400/h/600/q/85
```

The following figure shows the generated thumbnail.

Notes

To prevent unauthorized users from accessing or downloading the input image by using a URL that does not carry any processing parameter, you can add the processing parameters to the request signature. The following is a simple example. For a detailed calculation method, see Request Signature.

http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-Signature=<signature-value>&watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWl5aW4uanBn/gravity/southeast

List of Supported Fonts

The following table lists fonts supported by CI.

Font Name	Supported Language
simfang.ttf	Chinese/English
Simhei.ttf	Chinese/English
simkai.ttf	Chinese/English
Simsun.ttc	Chinese/English
STHeiti Light.ttc	Chinese/English
STHeiti Medium.ttc	Chinese/English
Youyuan.TTF	Chinese/English
ahronbd.ttf	English
arial.ttf	English
ariblk.ttf	English
Helvetica.dfont	English
HelveticaNeue.dfont	English

Text Watermark

Feature Overview

CI provides real-time text watermark processing through the watermark API. The size of the input image cannot exceed 32 MB, its width and height cannot exceed 30,000 pixels, and the total number of pixels cannot exceed 250 million. The width and height of the output image cannot exceed 9,999 pixels. For animated images, the product of width, height, and FPS cannot exceed 250 million pixels.

This feature supports the following processing methods.

- Processing upon download

API Example

1. Processing upon Download

```
GET /<ObjectKey>?watermark/2/text/<encodedText>
                /font/<encodedFont>
                /fontsize/<fontSize>
                /fill/<encodedColor>
                /dissolve/<dissolve>
                /gravity/<gravity>
                /dx/<dx>
                /dy/<dy>
                /batch/<type>
                /degree/<degree>
                /shadow/<shadow> HTTP/1.1
Host: <BucketName>.<Endpoint>
Date: <GMT Date>
Authorization: <Auth String>
```

> Note:
>
> - Spaces and line breaks in the preceding figure can be ignored.
> - Authorization: Auth String (For details, please refer to section Request Signature).

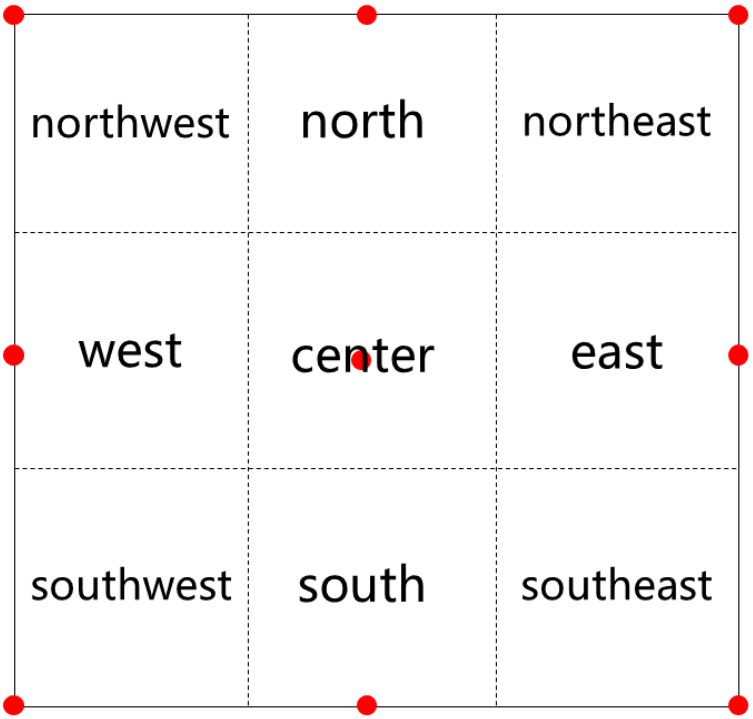
Processing Parameter Description

Operation name: watermark. The number 2 indicates that the watermark type is text.

Parameter	Description
ObjectKey	File name of the object, for example, folder/sample.jpg.
/text/	URL-safe Base64-encoded watermark content.
/font/	URL-safe Base64-encoded watermark font. The default value is Tahoma.ttf. For a list of supported fonts, see List of Supported Fonts.
/fontsize/	Font size, in pt. Default value: 13. To scale the text watermark proportionally based on the original image, convert the text watermark to a PNG image. For details, see Image Watermarking.
/fill/	Font color. The default font color is gray, and it needs to be set to hexadecimal RGB format (for example, #FF0000). For details, see RGB Color Table. The value should be URL-safe Base64-encoded, and the default value is #3D3D3D.
/dissolve/	Text transparency. The value range is 1–100. The default value is 90 (90% opacity).
/gravity/	Position of the text watermark in a Sudoku grid layout (See Sudoku Orientation Diagram)). The default value is SouthEast.
/dx/	Horizontal (X-axis) margin, in pixels. The default value is 0.
/dy/	Vertical (Y-axis) margin, in pixels. The default value is 0.
/batch/	Watermark tiling feature, which allows the text watermark to be tiled over the entire image. When the value is 1, the watermark tiling feature is enabled.
/degree/	Rotation angle of the text watermark. It is valid only when batch is set to 1. The value range is 0 to 360, and the default value is 0.
/shadow/	Text shadow effect. Value range: [0,100]. The default value is 0, indicating no shadow.

Sudoku Orientation Diagram

The Sudoku orientation diagram provides reference positions for various operations on the image. The red dot represents the origin of each region (After selecting a region through the gravity parameter, displacement operations refer to the corresponding origin).



> Note:
>
> - When the gravity parameter is set to center, the dx and dy parameters are invalid.
> - When the gravity parameter is set to north or south, the dx parameter is invalid (the watermark is horizontally centered).
> - When the gravity parameter is set to west or east, the dy parameter is invalid (the watermark is vertically centered).

Practical Case

> Note:
>
> This type of processing does not save the output image to the bucket.

Example 1: Adding a Text Watermark

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?watermark/2/text/6IW-6K6v5LqRwrfkuIfosaHkvJjlm74/fill/IzNEM0QzRA/fontsize/20/dissolve/50/gravity/northeast/dx/20/dy/20/batch/1/degree/45
```

The following figure shows the effect after a text watermark is added.

Notes

To prevent unauthorized users from accessing or downloading the input image by using a URL that does not carry any processing parameter, you can add the processing parameters to the request signature. The following is a simple example. For a detailed calculation method, see Request Signature.

http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-Signature=<signature-value>&watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWl5aW4uanBn/gravity/southeast

Rotation

Feature Overview

CI provides the rotation feature through the imageMogr2 API, including standard rotation and adaptive rotation.

This feature supports the following processing methods.

- Processing upon download

Restrictions

- Supported formats: JPG, BMP, GIF, PNG, and WebP images can be processed, and HEIF images can be decoded and processed.
- Size limit: The size of the input image cannot be larger than 32 MB, its width and height cannot exceed 30,000 pixels, and the total number of pixels cannot exceed 250 million. The width and height of the output image cannot exceed 9,999 pixels. For an animated image, the product of width x height x FPS cannot exceed 250 million.
- FPS (for animated images): For GIFs, the FPS cannot exceed 300.

API Form

1. Processing upon Download

```
GET /<ObjectKey>?imageMogr2/rotate/<rotateDegree>
          /auto-orient HTTP/1.1
Host: <BucketName>.<Endpoint>
Date: <GMT Date>
Authorization: <Auth String>
```

> Note:
>
> - Spaces and line breaks in the preceding figure can be ignored.
> - Authorization: Auth String (For details, please refer to section Request Signature).

Processing Parameter Description

Parameter	Description
ObjectKey	File name of the object, for example, folder/sample.jpg.
/rotate/<rotateDegree>	Standard rotation: Rotates an image clockwise by a specified angle. Value range: 0–360. By default, the image is not rotated.
/auto-orient	Adaptive rotation: Rotates an image adaptively based on its EXIF information.
/flip/<flip>	Mirror flip: The value vertical indicates vertical flipping, and the value horizontal indicates horizontal flipping.
/ignore-error/1	If this parameter is carried and the image fails to be processed because the image is too large or a parameter value exceeds the limit, the input image is returned and no error is reported.

Practical Case

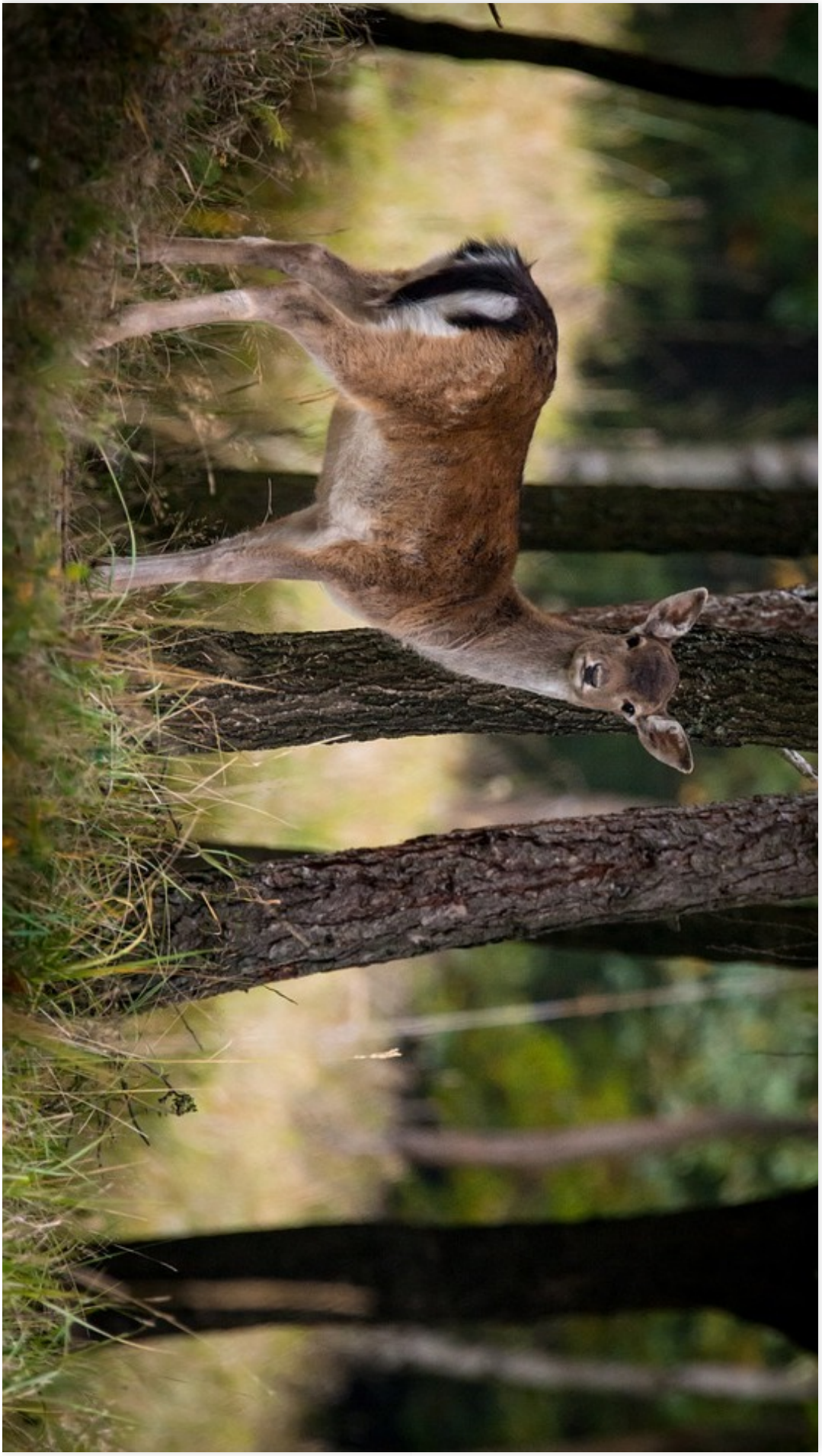
> Note:
>
> This type of processing does not save the output image to the bucket.

Example 1: Standard Rotation

The following example rotates an image clockwise by 90 degrees.

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/rotate/90
```

The following figure shows the final effect.



Notes

To prevent unauthorized users from accessing or downloading the input image by using a URL that does not carry any processing parameter, you can add the processing parameters to the request signature. The following is a simple example. For a detailed calculation method, see Request Signature.

http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-Signature=<signature-value>&watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWl5aW4uanBn/gravity/southeast

Format Conversion

Feature Overview

CI provides the format conversion, GIF optimization, and progressive display features through the imageMogr2 API.

This feature supports the following processing methods.

- Processing upon download

Restrictions

- Conversion between GIF and WebP formats is supported.
- Conversion between JPG, PNG, and BMP formats is supported.

API Form

1. Processing upon Download

```
GET /<ObjectKey>?imageMogr2/format/<Format>
            /cgif/<FrameNumber>
            /interlace/<Mode> HTTP/1.1
Host: <BucketName>.<Endpoint>
Date: <GMT Date>
Authorization: <Auth String>
```

> Note:
>
> Authorization: Auth String (For details, please refer to section Request Signature).

Processing Parameter Description

Parameter	Description
ObjectKey	File name of the object, for example, folder/sample.jpg.

Parameter	Description
/format/<Format>	Format conversion. The target thumbnail format can be JPG, BMP, GIF, PNG, WebP, or YJPEG (which is optimized by CI based on JPEG and is essentially JPG). The default format is the format of the input image.
/cgif/<FrameNumber>	GIF optimization. It only applies to input images in GIF format and optimizes a GIF image by reducing the number of frames and colors. It includes the following scenarios. - If FrameNumber = 1, it is processed based on the default FPS of 30. If the image FPS is greater than 30, it will be truncated. - If FrameNumber is in the range of (1,100], the image is compressed to the specified FPS (FrameNumber).
/interlace/<Mode>	Progressive mode of the output JPG image. Valid values: 0 and 1. 0: Progressive is not enabled. 1: Progressive is enabled. This parameter is valid only when the output image format is JPG. If the output image is not in JPG format, this parameter is ignored and the default value is 0.
/ignore-error/1	If this parameter is carried and the image fails to be processed because the image is too large or a parameter value exceeds the limit, the input image is returned and no error is reported.

Practical Case

> Note:
>
> This type of processing does not save the output image to the bucket.

Example 1: Converting a JPEG Image into a PNG Image

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/format/png
```

Notes

To prevent unauthorized users from accessing or downloading the input image by using a URL that does not carry any processing parameter, you can add the processing parameters to the request signature. The following is a simple example. For a detailed calculation method, see Request Signature.

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Am
```

z-Signature=<signature-value>&watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWl5aW4uanBn/gravity/southeast

Grayscale Image

Feature Overview

CI sets an image to a grayscale image through the imageMogr2/grayscale API.

API Example

1. Processing upon Download

```
GET /<ObjectKey>?imageMogr2/grayscale/<value> HTTP/1.1
Host: <BucketName>.<Endpoint>
Date: <GMT Date>
Authorization: <Auth String>
```

> Note:
>
> Authorization: Auth String (For details, please refer to section Request Signature).

Processing Parameter Description

Operation name: grayscale.

Parameter	Description
ObjectKey	File name of the object, for example, folder/sample.jpg.
/grayscale/<value>	Sets an image to be a grayscale image. The value 0 indicates that the image remains unchanged. The value 1 indicates that the image is set to a grayscale image.
/ignore-error/1	If this parameter is carried and the image fails to be processed because the image is too large or a parameter value exceeds the limit, the input image is returned and no error is reported.

Practical Case

> Note:
>
> This type of processing does not save the output image to the bucket.

Sample 1: Setting an Image to a Grayscale Image

The following example sets an image to a grayscale image.

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/grayscale/1
```

The following figure shows the final effect.

Notes

To prevent unauthorized users from accessing or downloading the input image by using a URL that does not carry any processing parameter, you can add the processing parameters to the request signature. The following is a simple example. For a detailed calculation method, see Request Signature.

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-Signature=<signature-value>&watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWl5aW4uanBn/gravity/southeast
```

Pipeline Operator

Feature Overview

The pipeline operator ｜ of CI ensures multiple operations on an image in sequence. Users can separate multiple processing parameters with the pipeline operator, allowing for different operations on an image in a single access. The size of the input image cannot exceed 32 MB, its width and height cannot exceed 30,000 pixels, and the total number of pixels cannot exceed 250 million. The width and height of the output image cannot exceed 9,999 pixels. For animated images, the product of width x height x FPS cannot exceed 250 million pixels.

API Example

Users can connect the processing styles to the image URL with a style separator ？ and separate multiple processing styles with the pipeline operator ｜. Styles are executed in sequence, and up to 10 pipeline layers are supported.

Practical Case

In this example, the input image is first scaled down to 50%, and then the text watermark "CI" is added to the lower right corner of the image.

It is assumed that the URL of the input image is as follows.

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg
```

The following figure shows the input image.

Example 1: Scaling the Image and Adding a Watermark

Add the scaling operation. The URL is as follows:

http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/thumbnail/!50p

The following figure shows the scaled image.

Append a pipeline operator (|) and add text watermark style processing. The final URL is as follows:

http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/thumbnail/!50p|watermark/2/text/5pWw5o2u5LiH6LGh/fill/I0ZGRkZGRg==/fontsize/30/dx/20/dy/20

The following figure shows the final output image.

Notes

To prevent unauthorized users from accessing or downloading the input image by using a URL that does not carry any processing parameter, you can add the processing parameters to the request signature. The following is a simple example. For a detailed calculation method, see Request Signature.

http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-Signature=<signature-value>&watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWl5aW4uanBn/gravity/southeast

Scaling

Feature Overview

CI provides the image scaling feature through the imageMogr2 API. The size of the input image cannot exceed 32 MB, its width and height cannot exceed 30,000 pixels, and the total number of pixels cannot exceed 250 million. The width and height of the output image cannot exceed 9,999 pixels. For animated images, the product of width, height, and FPS cannot exceed 250 million pixels.

This feature supports the following processing methods.

- Processing upon download

API Example

1. Processing upon Download

```
GET /<ObjectKey>?imageMogr2/thumbnail/<imageSizeAndOffsetGeometry> HTTP/1.1
Host: <BucketName>.<Endpoint>
Date: <GMT Date>
Authorization: <Auth String>
```

> Note:
>
> Authorization: Auth String (For details, refer to Request Signature).

Processing Parameter Description

Operation name: thumbnail.

Parameter	Description
ObjectKey	File name of the object, for example, folder/sample.jpg.
/thumbnail/!<Scale>p	Percentage to scale the width and height of the input image.
/thumbnail/!<Scale>px	Percentage to scale the width of the input image, without changing the height.
/thumbnail/!<Scale>p	Percentage to scale the height of the input image, without changing the width.

Parameter	Description
/thumbnail/<Width>x	Width of the output image, with the height scaled proportionally.
/thumbnail/x<Height>	Height of the output image, with the width scaled proportionally.
/thumbnail/<Width>x<Height>	Limits the maximum width and height of the thumbnail to Width and Height, respectively, for proportional scaling.
/thumbnail/<Width>x<Height>``>	Limits the maximum width and height of the thumbnail to Width and Height, respectively, for proportional scaling down. The smaller value between the width scaling ratio and height scaling ratio is used. If both the target width and height are larger than the original width and height, the image remains unchanged.
/thumbnail/<Width>x<Height>``<	Limits the maximum width and height of the thumbnail to Width and Height, respectively, for proportional scaling up. The smaller value between the width scaling ratio and height scaling ratio is used. If both the target width and height are smaller than the original width and height, the image remains unchanged.
/thumbnail/!<Width>x<Height>r	Limits the minimum width and height of the thumbnail to Width and Height, respectively, for proportional scaling.
/thumbnail/<Width>x<Height>!	Ignores the original aspect ratio, specifies the image width as Width and height as Height, and forcibly scales the image, which may cause the output image to be distorted.
/thumbnail/<Area>@	Scales the image proportionally. The total number of pixels of the output image cannot exceed Area.
/pad/	Whether to pad the blank area with the color specified by the color parameter after the input image is scaled as large as possible in a rectangle with the specified width and height, with the image centered. Valid values: 0 (not to pad) and 1 (to pad).
/color/	Padding color, which is white by default. It needs to be set in hexadecimal RGB format (for example, #FF0000). For details, see RGB Color Table. It should be URL-safe Base64-encoded, and the default value is #FFFFFF.
/ignore-error/1	If this parameter is carried and the image fails to be processed because the image is too large or a parameter value exceeds the limit, the input image is returned and no error is reported.

Practical Case

> Note:
>
> This type of processing does not save the output image to the bucket.

The input image is as follows.

Example 1: Scaling the Width and Height

The following example scales the image to 50% of its original width and height.

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/thumbnail/!50p
```

The following figure shows the final effect.

Example 2: Scaling the Width Without Changing the Height

The following example scales down the width of the input image by 50%, without changing its height.

http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/thumbnail/!50px

The following figure shows the final effect.

Example 3: Scaling in Padding Mode

The following example scales the input image as large as possible in a 600 x 600 rectangle and pads the blank area with a specified color.

http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/thumbnail/600x600/pad/1/color/IzNEM0QzRA

The following figure shows the final effect.

Notes

To prevent unauthorized users from accessing or downloading the input image by using a URL that does not carry any processing parameter, you can add the processing parameters to the request signature. The following is a simple example. For a detailed calculation method, see Request Signature.

http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-Signature=<signature-value>&watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWl5aW4uanBn/gravity/southeast

Cropping

Feature Overview

CI provides a cropping feature through the imageMogr2 API, including standard cropping, zoom cropping, inscribed circle cropping, and rounded corner cropping.

This feature supports the following processing methods.

- Processing upon download

Restrictions

- Supported formats: JPG, BMP, GIF, PNG, and WebP images can be processed, and HEIF images can be decoded and processed.
- Size limit: The size of the input image cannot be larger than 32 MB, its width and height cannot exceed 30,000 pixels, and the total number of pixels cannot exceed 250 million. The width and height of the output image cannot exceed 9,999 pixels. For an animated image, the product of width x height x FPS cannot exceed 250 million.
- FPS (for animated images): For GIFs, the FPS cannot exceed 300.

API Example

1. Processing upon Download

```
GET /<ObjectKey>?imageMogr2/cut/<width>x<height>x<dx>x<dy>
                        /crop/<imageSizeAndOffsetGeometry>
                        /iradius/<radius>
                        /rradius/<radius>
                        /scrop/<Width>x<Height> HTTP/1.1
Host: <BucketName>.<Endpoint>
Date: <GMT Date>
Authorization: <Auth String>
```

> Note:
>
> - Spaces and line breaks in the preceding figure can be ignored.
> - Authorization: Auth String (For details, refer to Request Signature).

Processing Parameter Description

Parameter	Description
ObjectKey	File name of the object, for example, folder/sample.jpg.
/ignore-error/1	If this parameter is carried and the image fails to be processed because the image is too large or a parameter value exceeds the limit, the input image is returned and no error is reported.

Parameters for Standard Cropping

Operation name: cut.

Parameter	Description
<width>	Sets the output image width to width.
<height>	Sets the output image height to height.
<dx>	Offsets horizontally to the right from the top left vertex of the image by dx.
<dy>	Offsets vertically downward from the top left vertex of the image by dy.

> Note:
>
> The parameter value should be greater than 0 and smaller than the width/height of the input image.

Parameters for Zoom Cropping

Operation name: crop.

Parameter	Description
/crop/<Width>x	Sets the output image width to Width, with the height unchanged. The value of Width should be greater than 0 and less than the width of the input image.
/crop/x<Height>	Sets the output image height to Height, with the width unchanged. The value of Height should be greater than 0 and less than the height of the input image.
/crop/<Width>x<Height>	Sets the output image width to Width and height to Height. The values of both Width and Height should be greater than 0 and less than the width/height of the input image.

During zoom cropping, you can also specify the starting point by using the gravity parameter. For details,

see Zoom Cropping Example.

Parameters for Inscribed Circle Cropping

Operation name: iradius.

Parameter	Description
/iradius/<radius>	Inscribed circle cropping feature, where radius is the radius of the inscribed circle. The value range is an integer greater than 0 and less than half of the shortest side of the input image. The center of the inscribed circle is the center of the image. This parameter is not supported when the image format is GIF.

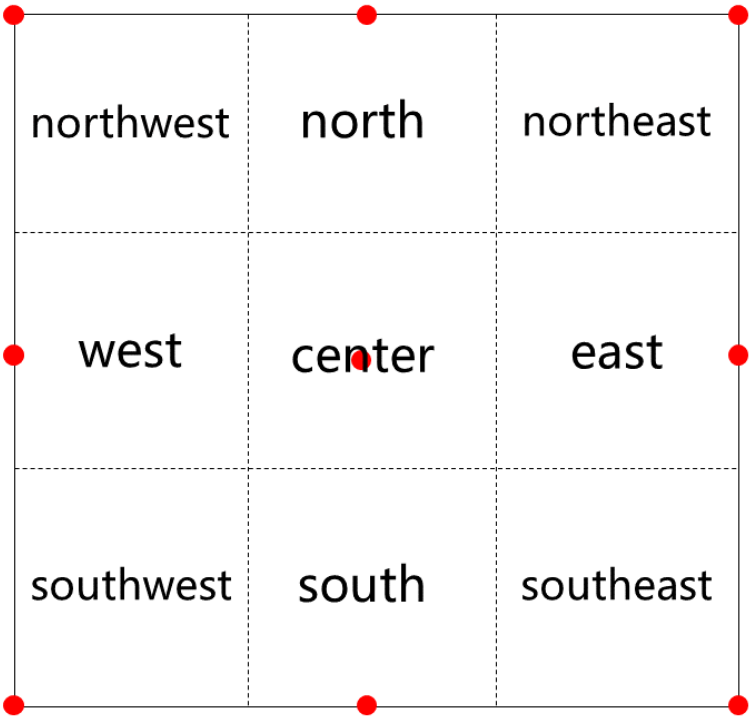
Parameters for Rounded Corner Cropping

Operation name: rradius.

Parameter	Description
/rradius/<radius>	Rounded corners cropping feature, where radius is the radius of the image's rounded edges. The value range is an integer greater than 0 and less than half of the shortest side of the input image. The rounded corners are tangent to the edges of the input image. This parameter is not supported when the image format is GIF.

Sudoku Orientation Diagram

The Sudoku orientation diagram provides reference positions for various operations on the image. The red dot represents the origin of each region (After selecting a region through the gravity parameter, displacement operations refer to the corresponding origin).



> Note:
>
> - When the gravity parameter is set to center, the dx and dy parameters are invalid.
> - When the gravity parameter is set to north or south, the dx parameter is invalid (the watermark is horizontally centered).
> - When the gravity parameter is set to west or east, the dy parameter is invalid (the watermark is vertically centered).

Practical Case

> Note:
>
> This type of processing does not save the output image to the bucket.

Example 1: Standard Cropping

The following example translates 100 pixels horizontally to the right and 10 pixels vertically to the bottom relative to the top-left vertex of an image to crop the image to 600 × 600.

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/cut/600x600x100x10
```

The input image is as follows.

The following figure shows the final effect.

Example 2: Zoom Cropping

The following example zoom and crop an image to 300 × 400 with the center point as a reference.

http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/crop/300x400/gravity/center

The following figure shows the final effect.

Example 3: Inscribed Circle Cropping

The following example performs inscribed circle cropping with a radius of 200.

http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/iradius/200

The following figure shows the final effect.

Example 4: Rounded Corner Cropping

The following example performs rounded corner cropping with a radius of 100.

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/rradius/100
```

The following figure shows the final effect.

Notes

To prevent unauthorized users from accessing or downloading the input image by using a URL that does not carry any processing parameter, you can add the processing parameters to the request signature. The following is a simple example. For a detailed calculation method, see Request Signature.

http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-Signature=<signature-value>&watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWl5aW4uanBn/gravity/southeast

Quality Transformation

Feature Overview

CI adjusts the quality of an image through the imageMogr2 API.

This feature supports the following processing methods.

- Processing upon download

> Note:
>
> This API supports images in JPG and WebP formats.

API Example

1. Processing upon Download

```
GET /<ObjectKey>?imageMogr2/quality/<Quality>
                        /rquality/<quality>
                        /lquality/<quality> HTTP/1.1
Host: <BucketName>.<Endpoint>
Date: <GMT Date>
Authorization: <Auth String>
```

> Note:
>
> Authorization: Auth String (For details, refer to Request Signature).

Processing Parameter Description

Operation name: quality.

Parameter	Description
ObjectKey	File name of the object, for example, folder/sample.jpg.
/quality/<Quality>	Absolute quality of the image. Value range: 0–100. Default value: input image quality. It takes the minimum value of the input image quality and the specified quality. Adding a "!" after means forced use of the specified value, for example, 90! .

Parameter	Description
/rquality/<quality>	Relative quality of the image. Value range: 0–100. The value is based on the input image quality. For example, if the input image quality is 80 and rquality is set to 80, the quality of the output image is 64 (80 x 80%).
/lquality/<quality>	Minimum quality of the image. Value range: 0–100. It indicates the minimum quality value of the output image. - For example, if the input image quality is 85 and lquality is set to 80, the quality of the output image is 85. - For example, if the input image quality is 60 and lquality is set to 80, the quality of the output image is improved to 80.
/ignore-error/1	If this parameter is carried and the image fails to be processed because the image is too large or a parameter value exceeds the limit, the input image is returned and no error is reported.

Practical Case

> Note:
>
> This type of processing does not save the output image to the bucket.

Example 1: Setting Absolute Quality

The following example sets the absolute quality to 60.

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/quality/60
```

The following figure shows the effect.

Example 2: Setting Relative Quality

The following example sets the relative quality to 60.

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/rquality/60
```

The following figure shows the effect.

Notes

To prevent unauthorized users from accessing or downloading the input image by using a URL that does not carry any processing parameter, you can add the processing parameters to the request signature. The following is a simple example. For a detailed calculation method, see Request Signature.

http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-Signature=<signature-value>&watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWl5aW4uanBn/gravity/southeast

Sharpening

Feature Overview

CI sharpens images through the imageMogr2 API.

This feature supports the following processing methods.

- Processing upon download

Restrictions

- Supported formats: JPG, BMP, GIF, PNG, and WebP images can be processed, and HEIF images can be decoded and processed.
- Size limit: The size of the input image cannot be larger than 32 MB, its width and height cannot exceed 30,000 pixels, and the total number of pixels cannot exceed 250 million. The width and height of the output image cannot exceed 9,999 pixels. For an animated image, the product of width x height x FPS cannot exceed 250 million.
- FPS (for animated images): For GIFs, the FPS cannot exceed 300.

API Example

1. Processing upon Download

```
GET /<ObjectKey>?imageMogr2/sharpen/<value> HTTP/1.1
Host: <BucketName>.<Endpoint>
Date: <GMT Date>
Authorization: <Auth String>
```

> Note:
>
> Authorization: Auth String (For details, refer to Request Signature).

Processing Parameter Description

Operation name: sharpen.

Parameter	Description

Parameter	Description
ObjectKey	File name of the object, for example, folder/sample.jpg.
/sharpen/<value>	Image sharpening feature. value is the sharpening parameter value, which is an integer in the range of 10 to 300 (recommended value: 70). A larger parameter value indicates a more pronounced sharpening effect.
/ignore-error/1	If this parameter is carried and the image fails to be processed because the image is too large or a parameter value exceeds the limit, the input image is returned and no error is reported.

Practical Case

> Note:
>
> This type of processing does not save the output image to the bucket.

Example 1: Sharpening

The following example sharpens an image with the sharpening parameter set to 70.

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/sharpen/70
```

The following figure shows the final effect.

Notes

To prevent unauthorized users from accessing or downloading the input image by using a URL that does not carry any processing parameter, you can add the processing parameters to the request signature. The following is a simple example. For a detailed calculation method, see Request Signature.

> http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-Signature=<signature-value>&watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWl5aW4uanBn/gravity/southeast

Limiting the Image Size

Feature Overview

CI limits the image size after processing (such as scaling and compression) through the imageMogr2/size-limit API.

Restrictions

- Supported formats: This API only supports JPG images.
- Size limits: The size of the input image cannot exceed 32 MB, its width and height cannot exceed 30,000 pixels, and the total number of pixels cannot exceed 250 million. The width and height of the output image cannot exceed 9,999 pixels.

API Example

```
GET /<ObjectKey>?imageMogr2/size-limit HTTP/1.1
Host: <BucketName>.<Endpoint>
Date: <GMT Date>
Authorization: <Auth String>
```

> Note:
>
> Authorization: Auth String (For details, refer to Request Signature).

Processing Parameter Description

Operation name: size-limit.

Parameter	Description
ObjectKey	File name of the object, for example, folder/sample.jpg.
size-limit	Limits the size of the image after conversion, supporting using megabytes (MB) and kilobytes (KB) as units. 1. Only JPG images are supported. It can be used to limit the size of the output image. 1. If ！ is added at the end, the size of the output image is compared with that of the input image. If the output image is smaller than the input image, the output image is

Parameter	Description
	returned. Otherwise, the input image is returned. For example: examplebucket-1250000000.cos.city.yfm4.fsphere.cn/picture.jpg?imageMogr2/size-limit/15k! 1. It is recommended that the parameter be used together with the strip parameter to remove some redundant information from the image for better effects. For example: examplebucket-1250000000.cos.city.yfm4.fsphere.cn/picture.jpg?imageMogr2/strip/format/png/size-limit/15k!.
/ignore-error/1	If this parameter is carried and the image fails to be processed because the image is too large or a parameter value exceeds the limit, the input image is returned and no error is reported.

Practical Case

> Note:
>
> This type of processing does not save the output image to the bucket.

Example 1: Converting the Image Format and Limiting the Output Image Size

The following example converts a JPG image into a PNG image and limits the output image size to 15 KB.

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/strip/format/png/size-limit/15k!
```

Notes

To prevent unauthorized users from accessing or downloading the input image by using a URL that does not carry any processing parameter, you can add the processing parameters to the request signature. The following is a simple example. For a detailed calculation method, see Request Signature.

```
http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-Signature=<signature-value>&watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWl5aW4uanBn/gravity/southeast
```

Gaussian Blurring

Feature Overview

CI blurs images through the imageMogr2 API.

This feature supports the following processing methods.

- Processing upon download

Restrictions

- Supported formats: JPG, BMP, GIF, PNG, and WebP images can be processed, and HEIF images can be decoded and processed.
- Size limit: The size of the input image cannot be larger than 32 MB, its width and height cannot exceed 30,000 pixels, and the total number of pixels cannot exceed 250 million. The width and height of the output image cannot exceed 9,999 pixels. For an animated image, the product of width x height x FPS cannot exceed 250 million.
- FPS (for animated images): For GIFs, the FPS cannot exceed 300.

API Example

1. Processing upon Download

```
GET /<ObjectKey>?imageMogr2/blur/<radius>x<sigma>  HTTP/1.1
Host: <BucketName>.<Endpoint>
Date: <GMT Date>
Authorization: <Auth String>
```

> Note:
>
> Authorization: Auth String (For details, refer to Request Signature).

Processing Parameter Description

Operation name: blur.

Parameter	Description

Parameter	Description
ObjectKey	File name of the object, for example, folder/sample.jpg.
radius<radius>	Blurring radius. Value range: 1–50
sigma<sigma>	Standard deviation of normal distribution, which should be greater than 0.
/ignore-error/1	If this parameter is carried and the image fails to be processed because the image is too large or a parameter value exceeds the limit, the input image is returned and no error is reported.

Practical Case

> Note:
>
> This type of processing does not save the output image to the bucket.

Example 1: Gaussian Blurring with a Radius of 8 and Sigma of 5

http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?imageMogr2/blur/8x5

The following figure shows the image after Gaussian blurring.

Notes

To prevent unauthorized users from accessing or downloading the input image by using a URL that does not carry any processing parameter, you can add the processing parameters to the request signature. The following is a simple example. For a detailed calculation method, see Request Signature.

http://examples-1251000004.cos.city.yfm4.fsphere.cn/sample.jpeg?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-Signature=<signature-value>&watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWl5aW4uanBn/gravity/southeast

Basic Image Compression (WebP)

Feature Description

Image compression refers to reducing the image size as much as possible while maintaining the original image quality, thereby saving image storage space, reducing image access traffic, and improving image access speed.
The WebP compression feature can convert images into the WebP compressed format, which is superior to JPG in compression performance. At the same image quality level, WebP images are over 25% smaller than JPG images, enabling adaptation to multi-terminal use cases.

API Examples

Processing upon Download

```
GET /<ObjectKey>?imageMogr2/format/webp HTTP/1.1
Host: <BucketName>.<Endpoint>
Date: <GMT Date>
Authorization: <Auth String>
```

Note

> - Authorization: A request header carrying authentication information to verify the validity of the request. The Pic-Operations header should be signed within the signature information. For details, see Request Signature.
> - When the fileid is the same as the ObjectKey, the output image will overwrite the input image for storage; otherwise, both the input and output images will be stored separately.
> - The value of fileid needs to be processed through URLEncode.

Processing Parameter Description

Parameter	Description	Type	Required
ObjectKey	File name of the object. For example, folder/sample.jpg.	String	Yes
/format/	Compression format (here is WebP).	String	Yes

Practical Cases

> Note:
>
> The practical cases in this document only include the Processing upon Download scenario. This type of processing does not save the output images to the bucket.

Assume the input image is in PNG format with a size of 1335.2 KB, as shown in the figure below.

Convert the input image to WebP format. The request URL is as follows.

```
http://example-1258125638.cos.city.yfm4.fsphere.cn/sample.png?imageMogr2/format/webp
```

The following figure shows the effect.

Compression Ratio Comparison

Format	Image Size
PNG (input image)	1335.2KB
WebP	65 KB (compression ratio: 95.13%)

FAQs
General

What Is COS?

COS is a cloud-based, non-hierarchical distributed storage product that provides you with cost-effective, fast, and reliable data storage solutions. COS stores your data across multiple availability zones (AZs) in a redundant manner and allows multiple clients or application threads to read or write to the data at the same time.
You can use web APIs to store and search for data through Cloud Virtual Machine (CVM) instances or over the Internet. You can also use the URL of a specified domain name to store and search for the content of each individual data object in COS through HTTP/HTTPS protocol.

What Is the Difference Between COS and Cloud File Storage (CFS)?

COS has no limit on directory hierarchy or data format and can store any amount of data. There is no upper limit on the storage capacity of buckets where no partitioning is required. It supports High Availability (HA) architecture deployment to ensure the eventual consistency of data but does not have features such as file lock. Its APIs can be accessed with the HTTP or HTTPS protocol, and its SDKs and tools can be integrated into your business. Objects uploaded to COS can be accessed or downloaded directly through URLs.
CFS uses common network file transfer protocols and can create file systems and implement large-scale expansion, but needs to be mounted onto CVM for use. It can store data for a wide range of applications such as websites, online distribution, and archiving. Featuring high computing throughput and extremely high availability and persistence, it is also suitable for scenarios demanding high concurrency or shared storage.

What Is the Difference Between COS and Cloud Block Storage (CBS)?

COS has no limits on file systems, directory structure, number of files, and storage capacity, and its storage needs to be managed and accessed via web APIs. It offers various SDKs and tools for integration, which can also be used separately without CVM. COS supports access to massive amounts of data but is not suitable for scenarios involving millisecond-level response or random read/write.
CBS needs to be used together with CVM and can only be mounted for use after the file system is partitioned or formatted. Based on its different types, it offers products with different performance metrics such as IOPS and throughput for different stand-alone usage scenarios.

Why Does the Access Link of a Public Read File Expire?

First, you can get the object address and the signed URL on the object detail page in the COS console.
If your file is public read:

- If you want your file to be always accessible to others, it is recommended that you directly use the

object address.
- If you only want your file to be accessible to others for a specific period, it is recommended that you directly replicate the signed URL, which carries the signature parameters and is valid for 1 hour.
  If your file is private:
- If you want your file to be always accessible to others, it is recommended that you change the file access permission to public read/private write and use the object address.
- If you want your file to be accessible to others for a specific period, it is recommended that you directly replicate the signed URL, which carries the signature parameters and is valid for 1 hour.

What Is a "Folder" or "Directory" in COS?

The concepts of folder and directory do not apply to COS. However, to accommodate the usage habits of different users, COS simulates the display of "folders" in the console with reference to the directory structure of traditional file management. For more information, see the document Folders and Directories.

Can COS Files Be Recovered After Being Deleted?

The data redundancy storage mechanism of COS is designed for scenarios where you need to recover data in case of faults in hardware such as servers. If you proactively and manually delete your data from COS or configure automated deletion, the data will be deleted as requested and cannot be recovered.
You can proactively delete files in the following ways:

- Use the COS console to delete a single file, delete files in batches, and clear incomplete multipart uploads or empty buckets.
- Use the COSCMD tool to delete files.
- Use COS APIs or SDKs to delete files.
- Use the COS lifecycle management feature to delete files regularly.
- Use the full sync feature of COS cross-region replication to synchronize the addition, modification, and deletion operations between buckets in different regions, so that existing files with the same name will be overwritten or deleted in the synchronized destination bucket.

How Can I Avoid Accidental Deletion?

- Back up the files in your bucket regularly in the following ways:
  - Use the cross-region replication feature to implement intra-region or cross-region bucket data backup.
  - Use COS APIs or SDKs regularly to back up your data to other COS buckets.
  - Use versioning to save the historical versions of your data.
- Use COS permission management. See Access Management Practices:
  - Separate read permission and write permission. For businesses where you only need to read data, only use a sub-account with read permission or a temporary key to access the data.
  - Separate permissions for different buckets. For different businesses, only authorize permissions

for buckets, directories, and operations within the scope of that particular business.
- Do not use a root account to access COS.
- Use a temporary key to access COS.
- Properly manage your data access secrets, such as the account password, CAM sub-account access secrets, and API key.

Does COS Support the Data Statistics Feature?

With the COS capability of monitoring stored data, you can learn the status and trends of all the data in the monitoring data window. To view general data trends, you can go to the overview page in the COS console to view data such as the storage size, number of requests, and traffic based on the dimensions of different storage types.
To view statistics of a single bucket, see Basic Data Statistics.
In addition, you can also view the monitoring information of different buckets on the Cloud Monitor page and configure different alarm policies according to your business needs.

Does COS Support Image Compression?

COS is a distributed storage service for unstructured data and does not support image compression.

Does COS Support Thumbnail Generation?

COS is a distributed storage service for unstructured data and does not support thumbnail generation.

Does COS Support Video Transcoding?

COS is a distributed storage service for unstructured data and does not support video transcoding.

Does COS Support Auto Decompression of Uploaded Files?

COS is a distributed storage service for unstructured data and does not support file decompression.

Billing

How Is COS Billed?

The cost of COS consists of three parts: storage fee, request fee, and traffic fee.

The billing method for COS is currently pay-as-you-go.

Billing Item	Billing Item Description	Billing Formula
COS Standard Storage Capacity	The billing is calculated based on the storage capacity, different storage types have different unit prices.	Storage capacity fee = storage capacity unit price x daily storage capacity.
COS Standard Storage Read Request	The request cost is calculated based on the number of requests, and the unit price of requests varies for different storage types.	Request fee = price per ten thousand requests x daily cumulative number of requests / 10000.
COS Standard Storage Write Request	The request cost is calculated based on the number of requests, and the unit price of requests varies for different storage types.	Request fee = price per ten thousand requests x daily cumulative number of requests / 10000.
COS Outbound External Network Traffic	Public network downstream traffic.	Traffic fee = price per GB x daily accumulated traffic.
COS Cross-region Replication Traffic	Cross-region Replication Traffic.	Traffic fee = price per GB x daily accumulated traffic.

How Do I View the Bills?

You can view the fees incurred by your account using the object storage service at the Billing Center in the console.

Does Transferring Data Between COS and CVM Incur Charges for Requests and Traffic?

Transferring data between COS and CVM in the same region is considered transmission in a private network, so the traffic is free. Transferring data between COS and CVM in different regions incurs traffic charges.

Requests generated by data transfer between COS and CVM, regardless of the region and whether they are over a private or public network, will be charged.

How Is Outbound Public Network Traffic from COS Billed?

Outbound public network traffic is the traffic generated by data transfer from COS to the client over the Internet. Traffic generated by downloading an object directly via an object link or by browsing an object via a static website access node is considered outbound public network traffic, and the corresponding fees are the outbound public network traffic fees.

Does Uploading Files to a COS Bucket Generate Traffic Fees?

No.

Does Accessing between Cloud Products in the Same Region Generate Traffic Fees?

When you access cloud products in the same region, a private network connection will be automatically used, which will not generate traffic fees.

Can I Still Access and Download Files in the COS Console If My COS Service Is Suspended Due to Overdue Payment?

After the COS service is suspended due to overdue payment, you cannot access the console. During this period, if you do not top up your account to make the balance greater than or equal to 0, your data will continue to be isolated.

Do Traffic Fees Incur If an Object Stored in COS Is Subject to DDoS Attacks?

Traffic fees do not incur if an object is subjected to DDoS attacks. COS buckets have a hotlink protection feature. You can reduce the impact of DDoS attacks after enabling the hotlink protection setting.

Upload and Download

Does COS Limit the Upload and Download Bandwidth?

COS does not limit the upload or download bandwidth. The specific upload and download speeds depend on your local bandwidth.

How Can I Directly Preview a File In My Browser without Downloading It?

1. If you wish to preview files directly through a browser or download objects of .apk or .ipa type in the bucket, access the objects through a custom domain name.
2. You can set the parameter value of Content-Disposition in the custom object headers to `inline` via the COS console. When the browser supports the file format, it will open the file directly instead of downloading it. For the operation guide, see Custom Headers.

How Do I Directly Download a File in My Browser Without Previewing It?

You can set the parameter value of Content-Disposition in the custom object headers to `attachment` via the COS console. For the operation guide, see Custom Headers.
You can also set the request parameter value of response-content-disposition in the GET Object API to `attachment` to enable your browser to pop up a window for downloading the file. For the reference documentation, see GET Object.

> Note:
>
> To use the response-* parameter in a request, the request must be signed.

How Do I Determine If I Am Accessing COS Over a Private Network?

Intelligent DNS resolution is adopted for COS access endpoints, and the optimal linkage will be detected and provided for you to access COS over the Internet with different Internet Service Providers (ISPs). If you have deployed a service within the cloud platform for accessing COS, intra-region access will be automatically routed to a private network address. Cross-region requests do not support private network access and will be resolved to a public network address by default.

Ways to Determine Access Over a Private Network

Cloud products within the same region automatically access each other through private network connections, and the generated private network traffic incurs no traffic fees. Therefore, when you purchase different products, it is recommended that you choose the same region to save on fees.
To determine access over a private network, see the following method:
For example, when a CVM accesses COS, to determine whether a private network is used to access COS, use the `nslookup` command on the CVM to resolve the COS endpoint. If a private IP address is returned, the access between the CVM and COS is over a private network; otherwise, it is over a public network.

> Note:
>
> Generally, a private IP address takes the form of 10.*.*.* or 100.*.*.* , and a VPC IP address takes the form of 169.254.*.* .

Assume that examplebucket-1250000000.cos.city.yfm4.fsphere.cn is the address of the destination bucket; the Address: 10.148.214.13 below indicates that the access is over a private network.

```
nslookup examplebucket-1250000000.cos.city.yfm4.fsphere.cn

Server:         10.138.224.65
Address:        10.138.224.65  #53

Name:   examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Address: 10.148.214.13
Name:   examplebucket-1250000000.cos.city.yfm4.fsphere.cn
Address: 10.148.214.14
```

What If the Error 403 Forbidden Occurs or Access Permission Is Rejected When I Perform Upload/Download and Other Operations?

Follow the steps below to troubleshoot the issue step by step:

1. Check if the following configuration information is correct:
   BucketName, APPID, Region, SecretId, and SecretKey.
2. After ensuring that the above information is correct, check if you are using a sub-account for operations. If you are, check whether the root account has granted the sub-account the permissions. Otherwise, log in to the root account to grant the sub-account the permissions. For operations on granting permissions, see Authorization Cases.
3. If you are using a temporary key for operations, check whether the current operation is within the policy set when you get the key.
4. If you still cannot solve the problem after following the steps above, submit a ticket and contact us.

When I Upload a New File to a Bucket in Which Another File of the Same Name Exists, Will the Old File Be Overwritten or Will the New File Be Saved with a Different Version Name?

COS now supports the versioning feature. If versioning is not enabled for the bucket, when you upload a file to the bucket in which another file of the same name already exists, the old file will be directly overwritten; if versioning is enabled for the bucket, when you upload files to the bucket in which another file of the same name already exists, multiple versions of the object will co-exist.

What Is the Minimum Part Size of a Multipart Upload in COS?

1 MB minimum per part. For more information, see the document Specifications and Limits.

When Uploading a Large File with Multipart Upload, Can I Replace an Invalid Signature to Continue the Multipart Upload?

Yes.

Data Security

Key-Related Issues

Where can I view key information such as APPID, SecretId, and SecretKey?

The second half of a bucket name is the APPID. You can view it by logging in to the COS console. To view information such as SecretId and SecretKey, log in to Cloud Access Management > Cloud API Key.

What should be done if key-related information such as APPID and SecretId, is leaked?

Users can delete the leaked keys and create new ones.

ACL+ Policy and Other Permission Issues

What if the error 403 forbidden occurs or access permission is rejected when I perform upload/download and other operations?

Follow the steps below to troubleshoot the issue step by step:

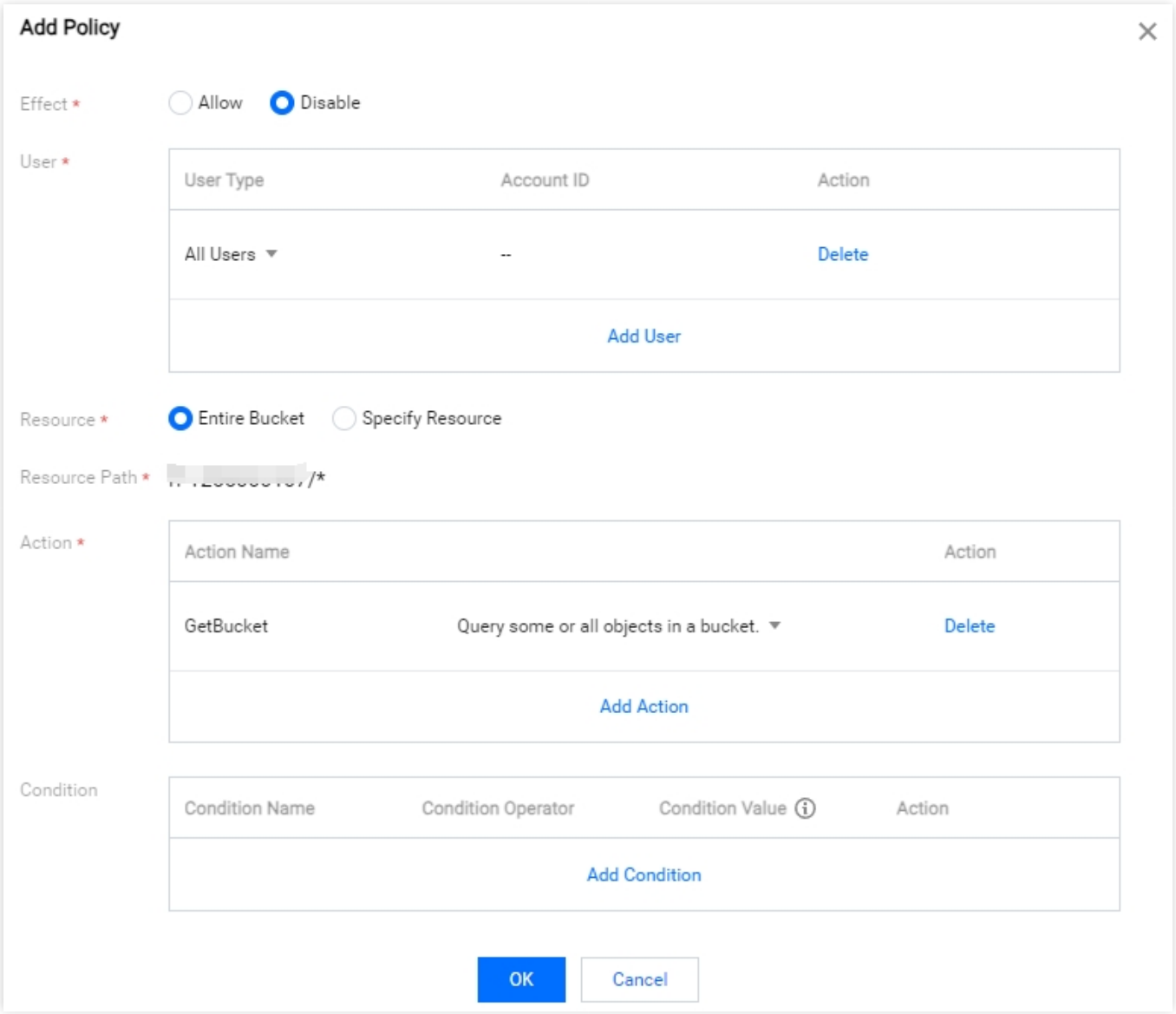
1. Check whether your following configuration information is correct:
   BucketName, APPID, Region, SecretId, SecretKey, etc.
2. After ensuring that the above information is correct, check whether you are using a sub-account for operation. If you are using a sub-account, check whether the root account has authorized the sub-account. Otherwise, you should log in to the root account first to authorize the sub-account.
   For more information on permissions, see Access Management Permission Setting Related Cases.
3. If you use a temporary key to perform an operation, check whether the current operation is within the policy set when obtaining the temporary key. Otherwise, you should modify the related Policy settings.

When accessing a public read bucket using the bucket's default domain, a file list is returned. How can I hide the file list information?

You can set a Get Bucket permission to deny anyone for the corresponding storage bucket. The steps are as follows:
Log in to the COS console, select the bucket list, and enter the Permission Management of the corresponding storage bucket page.

Method 1:

1. Find Policy Permission Settings, and click Add Policy under Graphical Settings.
2. Add the corresponding permission settings as shown in the figure below and click OK.



Method 2:

Find Policy Permission Settings, click Policy Syntax > Modify, and enter the following expressions:

```
{
  "Statement": [
    {
      "Action": [
        "name/cos:GetBucket",
        "name/cos:GetBucketObjectVersions"
      ],
      "Effect": "Deny",
      "Principal": {
        "qcs": [
          "qcs::cam::anyone:anyone"
        ]
      },
```

```
      "Resource": [
        "qcs::cos:ap-city:uid/1250000000:examplebucket-1250000000/*"
      ]
    }
  ],
  "version": "2.0"
}
```

> Note:
>
> Replace the relevant information in "qcs::cos:ap-city:uid/1250000000:examplebucket-1250000000/*" with the following:
>
> - "ap-city" is replaced with the region where your bucket is located.
> - "1250000000" is replaced with your APPID information.
> - "examplebucket-1250000000" is replaced with your bucket name.
>
> Among them, APPID is the second half of the bucket name. You can view the bucket name in the COS Console.

Is the ACL restriction on COS for buckets or accounts? Is it possible to specify permissions when uploading files?

ACL restrictions are for the account. It is not recommended to specify permissions when uploading files, as this may cause an error if the ACL + Policy exceeds 1,000.

How do I authorize collaborators to access a specified bucket?

A collaborator account is a special sub-account. For more information, see Access Policy Language Overview.

If multiple businesses need to operate on a bucket, is it possible to isolate permissions based on buckets or other dimensions?

Log in to the CAM console and enter the user management page. You can enable sub-accounts for different businesses and grant different authorization operations.

What if the error: your policy or acl has reached the limit (Status Code: 400; Error Code: PolicyFull) is displayed when uploading files or creating buckets?

The maximum quantities of the buckets and object ACL + Policy rules under each COS root account is 1,000. When the number of related ACLs or Policy sets is greater than 1,000, this error appears. Therefore, it is recommended that useless ACLs or Policies should be deleted.

> Note:
>
> We do not recommend using file-level ACLs or Policies. We recommend that when you call the API or SDK if you do not need to perform special ACL control on files, leave the ACL-related parameters (such as x-cos-acl and ACL) blank to keep the inherited storage bucket permissions.

How do I create a sub-account for a subsidiary or employee and authorize access permissions

to a specific bucket?

For more information, see Granting Sub-Accounts Access to COS to create sub-accounts and grant them permissions.

How can I grant certain sub-accounts permission to operate only a certain bucket?

If you want the sub-account to have only operation permissions for a specific bucket, you can use the sub-account to add a path.

How do I use account A to authorize account B to have write permission on the buckets under account A?

For more information, see ACL Access Control Practice and CAM Access Management Practice to grant permissions.

Anti-hotlinking Issues

Can I set up an allowlist to allow access, and also allow access when the browser opens the link alone?

When setting up anti-hotlinking, select allow empty referer, and you can access the link by opening it in a browser alone, even if you set up an allowlist.

The anti-hotlinking allowlist of the storage bucket test is set, and a.com is allowed to access. But why cannot the web player under a.com playback video files in the bucket test?

When using Windows Media Player, Flash Player, and other players to playback video links on a web page, the referer in the request is empty, resulting in failure to hit the allowlist. It is recommended that an empty referer be allowed when setting the allowlist.

Glossary

APPID

The APPID is one of the account identifiers for a cloud platform account and is used for associating cloud resources. After you successfully sign up for a cloud platform account, you will be automatically assigned an APPID by the system. Multiple projects can be created under one APPID. You can view the APPID via the Account Center console.

Bucket

A bucket is used for storing objects in COS. Multiple objects can be stored in one bucket.

Object

An object is the basic unit stored in COS.

SecretId

SecretId and SecretKey, collectively called the Cloud API key, are the security credentials needed for authentication when you access the cloud platform APIs. SecretId is used to identify the API caller. Multiple Cloud API keys can be created under one APPID. If you do not have a Cloud API key, you need to create one at Cloud Access Management > Cloud API Key. Otherwise, you cannot call any Cloud APIs.

SecretKey

SecretId and SecretKey, collectively called the Cloud API key, are the security credentials needed for authentication when you access the cloud platform APIs. SecretKey is the key used for signature string encryption and server-side signature string verification. Multiple Cloud API keys can be created under one APPID. If you do not have a Cloud API key, you need to create one at Cloud Access Management > Cloud API Key. Otherwise, you cannot call any Cloud APIs.

Project

A project is a virtual concept where users can establish multiple projects under one account, each managing different resources.

API Documentation
cos(cos)
version(2020-01-07)
API overview

API version

V3

Cross-Cloud Replication

Interface name	Interface function
CheckBucketCrossCloudReplicationDestination	CheckBucketCrossCloudReplicationDestination
GetBucketCrossCloudReplication	GetBucketCrossCloudReplication
PutBucketCrossCloudReplication	PutBucketCrossCloudReplication

CallingMethod signaturev1

Converge Cloud authenticates each access request, i.e. each request needs to include authentication information (Signature) in the common parameters to verify the identity of the requester.
The Signature is generated by the security credentials which include SecretId and SecretKey. If you don't have the security credentials yet, go to the API Key page to apply for them; otherwise, you cannot invoke the API.

1. Applying for Security Credentials

Before using the API for the first time, go to the API Key page to apply for security credentials.
Security credentials consist of SecretId and SecretKey:

- SecretId is used to identify the API requester.
- SecretKey is used to encrypt the signature string and verify it on the server.
- You must keep your security credentials private and avoid disclosure.

You can apply for the security credentials through the following steps:

1. Log in to the Converge Cloud Console.
2. Go to the Cloud API Key page.
3. On the API Key Management page, click Create Key to create a SecretId/SecretKey pair.

Note: Each account can have up to two pairs of SecretId/SecretKey.

2. Generating a Signature

With the SecretId and SecretKey, a signature can be generated. The following describes how to generate a signature:

Assume that the SecretId and SecretKey are:

- SecretId: AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE
- SecretKey: Gu5t9xGARNpq86cd98joQYCN3EXAMPLE

Note: This is just an example. For actual operations, please use your own SecretId and SecretKey.

Take the Cloud Virtual Machine's request to view the instance list (DescribeInstances) as an example. When

you invoke this API, the request parameters may be as follows:

Parameter name	Description	Parameter value
Action	Method name	DescribeInstances
SecretId	Key ID	AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE
Timestamp	Current timestamp	1465185768
Nonce	Random positive integer	11886
Region	Region where the instance is located	luzon
InstanceIds.0	ID of the instance to query	ins-09dx96dg
Offset	Offset	0
Limit	Allowed maximum output	20
Version	API version number	2017-03-12

2.1. Sorting Parameters

First, sort all the request parameters in an ascending lexicographical order (ASCII code) by their names. Notes: (1) Parameters are sorted by their names instead of their values; (2) The parameters are sorted based on ASCII code, not in an alphabetical order or by values. For example, InstanceIds.2 should be arranged after InstanceIds.12. You can complete the sorting process using a sorting function in a programming language, such as the ksort function in PHP. The parameters in the example are sorted as follows:

```
{
    'Action' : 'DescribeInstances',
    'InstanceIds.0' : 'ins-09dx96dg',
    'Limit' : 20,
    'Nonce' : 11886,
    'Offset' : 0,
    'Region' : 'luzon',
    'SecretId' : 'AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE',
    'Timestamp' : 1465185768,
    'Version': '2017-03-12',
}
```

When developing in another programming language, you can sort these sample parameters and it will work as long as you obtain the same results.

2.2. Concatenating a Request String

This step generates a request string.
Format the request parameters sorted in the previous step into the form of "parameter name"="parameter value". For example, for the Action parameter, its parameter name is "Action" and its parameter value is "DescribeInstances", so it will become Action=DescribeInstances after formatted.
Note: The "parameter value" is the original value but not the value after URL encoding.

Then, concatenate the formatted parameters with "&". The resulting request string is as follows:

```
Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=luzon&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Timestamp=1465185768&Version=2017-03-12
```

2.3. Concatenating the Signature Original String

This step generates a signature original string.
The signature original string consists of the following parameters:

1. HTTP method: POST and GET modes are supported, and GET is used here for the request. Please note that the method name should be in all capital letters.
2. Request host: the domain name of the request to view the list of instances (DescribeInstances) is cvm.convergecloud.com. The actual request domain name varies by the module to which the API belongs. For more information, see the instructions of the specific API.
3. Request path: The request path in the current version of Cloud API is fixed to / .
4. Request string: the request string generated in the previous step.

The concatenation rule of the signature original string is: Request method + request host + request path + ? + request string

The concatenation result of the example is:

```
GETcvm.convergecloud.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=luzon&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Timestamp=1465185768&Version=2017-03-12
```

2.4. Generating a Signature String

This step generates a signature string.
First, use the HMAC-SHA1 algorithm to sign thesignature original stringobtained in the previous step, and then encode the generated signature using Base64 to obtain the final signature.

The specific code is as follows with the PHP language being used as an example:

```
$secretKey = 'Gu5t9xGARNpq86cd98joQYCN3EXAMPLE';
$srcStr = 'GETcvm.convergecloud.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit
=20&Nonce=11886&Offset=0&Region=luzon&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPL
E&Timestamp=1465185768&Version=2017-03-12';
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));
echo $signStr;
```

The final signature is:

```
EliP9YW3pW28FpsEdkXt/+WcGeI=
```

When developing in another programming language, you can sign and verify the original in the example above and it works as long as you get the same results.

3. Encoding a Signature String

The generated signature string cannot be directly used as a request parameter and must be URL encoded.

For example, if the signature string generated in the previous step is EliP9YW3pW28FpsEdkXt/+WcGeI= , the final signature string request parameter (Signature) is EliP9YW3pW28FpsEdkXt%2f%2bWcGeI%3d, which will be used to generate the final request URL.

**Note: If your request method is GET, or the request method is POST and the Content-Type is application/x-www-form-urlencoded, then all the request parameter values need to be URL encoded (except the parameter key and the symbol of =) when sending the request. Non-ASCII characters need to be encoded with UTF-8 before URL encoding. **

Note: The network libraries of some programming languages automatically URL encode all parameters, in which case there is no need to URL encode the signature string; otherwise, two rounds of URL encoding will cause the signature to fail.

Note: Other parameter values also need to be encoded using RFC 3986. Use %XY in percent-encoding for special characters such as Chinese characters, where "X" and "Y" are hexadecimal characters (0-9 and uppercase A-F), and using lowercase will cause an error.

4. Signature Failure

The following situational error codes for signature failure may occur. Please resolve the errors accordingly.

Error code	Error description
AuthFailure.SignatureExpire	The signature is expired

Error code	Error description
AuthFailure.SecretIdNotFound	The key does not exist
AuthFailure.SignatureFailure	Signature error
AuthFailure.TokenFailure	Token error
AuthFailure.InvalidSecretId	Invalid key (not a Cloud API key type)

5. Signature Demo

When calling API 3.0, you are recommended to use the corresponding Converge Cloud SDK 3.0, Swhich encapsulates the signature process, enabling you to focus on only the specific APIs provided by the product when developing. See SDK Center for more information. Currently, the following programming languages are supported:

- Python
- Java
- PHP
- Go
- Node

To further explain the signing process, we will use a programming language to implement the process described above. The request domain name, API and parameter values in the sample are used here. This goal of this example is only to provide additional clarification for the signature process, please see the SDK for actual usage.

The final output URL might be: https://cvm.convergecloud.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=luzon&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Signature=EliP9YW3pW28FpsEdkXt%2F%2BWcGeI%3D&Timestamp=1465185768&Version=2017-03-12

Note: The key in the example is fictitious, and the timestamp is not the current time of the system, so if this URL is opened in the browser or called using commands such as curl, an authentication error will be returned: Signature expired. In order to get a URL that can work properly, you need to replace the SecretId and SecretKey in the example with your real credentials and use the current time of the system as the Timestamp.

Note: In the example below, even if you use the same programming language, the order of the parameters in the URL may be different for each execution. However, the order does not matter, as long as all the parameters are included in the URL and the signature is calculated correctly.

Note: The following code is only applicable to API 3.0. It cannot be directly used in other signature processes. Even with an older API, signature calculation errors may occur due to the differences in details. Please refer to the corresponding documentation.

Java

```
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.Random;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;

public class CloudAPIDemo {
    private final static String CHARSET = "UTF-8";

    public static String sign(String s, String key, String method) throws Exception {
        Mac mac = Mac.getInstance(method);
        SecretKeySpec secretKeySpec = new SecretKeySpec(key.getBytes(CHARSET), mac.getAlgorithm());
        mac.init(secretKeySpec);
        byte[] hash = mac.doFinal(s.getBytes(CHARSET));
        return DatatypeConverter.printBase64Binary(hash);
    }

    public static String getStringToSign(TreeMap<String, Object> params) {
        StringBuilder s2s = new StringBuilder("GETcvm.convergecloud.com/?");
        // When signing, the parameters need to be sorted in lexicographical order. TreeMap is used here
to guarantee the correct order.
        for (String k : params.keySet()) {
            s2s.append(k).append("=").append(params.get(k).toString()).append("&");
        }
        return s2s.toString().substring(0, s2s.length() - 1);
    }

    public static String getUrl(TreeMap<String, Object> params) throws UnsupportedEncodingException
{
        StringBuilder url = new StringBuilder("https://cvm.convergecloud.com/?");
        // There is no requirement for the order of the parameters in the actual request URL.
        for (String k : params.keySet()) {
            // The request string needs to be URL encoded. As the Key is all in English letters, only the valu
e is URL encoded here.
            url.append(k).append("=").append(URLEncoder.encode(params.get(k).toString(), CHARSET)).app
end("&");
        }
        return url.toString().substring(0, url.length() - 1);
```

```
    }

    public static void main(String[] args) throws Exception {
        TreeMap<String, Object> params = new TreeMap<String, Object>(); // TreeMap enables automati
c sorting
        // A random number should be used when actually calling, for example: params.put("Nonce", new
Random().nextInt(java.lang.Integer.MAX_VALUE));
        params.put("Nonce", 11886); // Common parameter
        // The current time of the system should be used when actually calling, for example:   params.put
("Timestamp", System.currentTimeMillis() / 1000);
        params.put("Timestamp", 1465185768); // Common parameter
        params.put("SecretId", "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"); // Common parameter
        params.put("Action", "DescribeInstances"); // Common parameter
        params.put("Version", "2017-03-12"); // Common parameter
        params.put("Region", "luzon"); // Common parameter
        params.put("Limit", 20); // Business parameter
        params.put("Offset", 0); // Business parameter
        params.put("InstanceIds.0", "ins-09dx96dg"); // Business parameter
        params.put("Signature", sign(getStringToSign(params), "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE
", "HmacSHA1")); // Common parameter
        System.out.println(getUrl(params));
    }
}
```

Python

Note: If running in a Python 2 environment, the following requests dependency package must be installed first: pip install requests .

```
# -*- coding: utf8 -*-
import base64
import hashlib
import hmac
import time

import requests

secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"

def get_string_to_sign(method, endpoint, params):
    s = method + endpoint + "/?"
    query_str = "&".join("%s=%s" % (k, params[k]) for k in sorted(params))
    return s + query_str

def sign_str(key, s, method):
```

```
    hmac_str = hmac.new(key.encode("utf8"), s.encode("utf8"), method).digest()
    return base64.b64encode(hmac_str)

if __name__ == '__main__':
    endpoint = "cvm.convergecloud.com"
    data = {
        'Action' : 'DescribeInstances',
        'InstanceIds.0' : 'ins-09dx96dg',
        'Limit' : 20,
        'Nonce' : 11886,
        'Offset' : 0,
        'Region' : 'luzon',
        'SecretId' : secret_id,
        'Timestamp' : 1465185768, # int(time.time())
        'Version': '2017-03-12'
    }
    s = get_string_to_sign("GET", endpoint, data)
    data["Signature"] = sign_str(secret_key, s, hashlib.sha1)
    print(data["Signature"])
    # n actual invocation would occur here, which may incur fees after success
    # resp = requests.get("https://" + endpoint, params=data)
    # print(resp.url)
```

signaturev3

Converge Cloud The API will authenticate each access request, meaning that each request must include signature information (Signature) in the public request parameters to verify the requester's identity.
The signature information is generated by the security credentials, which include SecretId and SecretKey . If the user does not have security credentials, please apply for them on the Cloud API Key page, otherwise they cannot call the Cloud API interfaces.

1. Apply for security credentials.

Before using the Cloud API for the first time, please go to the Cloud API Key page to apply for a security credential.
Security credentials include SecretId and SecretKey:

- SecretId The security credentials are used to identify the identity of the API caller.
- SecretKey The secret key is used to encrypt and sign the string for server-side verification of the signature.
- Users must strictly protect their security credentials and avoid disclosure them.

The specific steps to apply for security credentials are as follows:

1. Log in to Converge Cloud Administration Center Console
2. Go to the console page for cloud API keys
3. On the Cloud API Key page, click New to create a pair of SecretId/SecretKey

Note: Developer accounts can have up to two pairs SecretId / SecretKey.

2. TC3-HMAC-SHA256 signature method

Note: For GET method, only Content-Type: application/x-www-form-urlencoded protocol format is supported. For POST method, Content-Type: application/json and Content-Type: multipart/form-data are currently supported. By default, all business interfaces are supported in json format. Only specific business interfaces are supported in multipart format. In this case, the interface cannot be called in json format. Refer to the specific business interface document description.

The following takes Cloud Virtual Machine querying Region-1 instance list as an example to introduce the signature calculation process step by step. We only use two parameters for the query instance list: Limit and Offset, called using the GET method.

Assume that the SecretId and SecretKey of the user are AKIDz 8krbsJ 5yKBZQpn 74WFkmLPx 3EXAMPLE and Gu 5t9xGARNpq 86cd 98joQYCN3EXAMPLE, respectively.

2.1. splice specification request string

Concatenate CanonicalRequest strings in the following format:

```
CanonicalRequest =
  HTTPRequestMethod + '\n' +
  CanonicalURI + '\n' +
  CanonicalQueryString + '\n' +
  CanonicalHeaders + '\n' +
  SignedHeaders + '\n' +
  HashedRequestPayload
```

- HTTPRequestMethod: HTTP request methods (GET, POST), GET in this example;
- CanonicalURI: URI parameter, API 3.0 fixed to forward slash (/);
- CanonicalQueryString: Query string in the URL that initiates the HTTP request. For POST requests, it is fixed as an empty string. For GET requests, it is the question mark (?) in the URL. The following string content, this example value is: Limit=10&Offset=0. Note: CanonicalQueryString needs to be URL encoded.
- CanonicalHeaders: The header information of participating signature, at least including host and content-type, can also be added with customized header to improve the uniqueness and security of its own request. Splicing rules: 1) head key and value are uniformly converted to lowercase, and the first and last spaces are removed, and spliced according to the format of key:value\n;2) multiple heads are spliced according to the dictionary sorting of head key (lowercase). In this example: content-type: application/x-www-form-urlencoded\nhost: cvm.convergecloud.com\n
- SignedHeaders: The header information involved in the signature indicates which headers participate in the signature of this request, and the header content contained in CanonicalHeaders is one-to-one corresponding. content-type and host are required headers. constructed as: 1) The head key is uniformly converted to lowercase; 2) Multiple header keys (lowercase) are concatenated according to dictionary sorting and separated by semicolons (;). In this case: content-type;host
- HashedRequestPayload: The hash value of the request body, calculated as Lowercase(HexEncode(Hash.SHA256(RequestPayload))), SHA256 hash the entire payload of the HTTP request body, then hexadecimal encoding, and finally converting the encoding string to lowercase letters. Note: For GET requests, RequestPayload is fixed as an empty string. For POST requests, RequestPayload is the payload of the HTTP request body.

According to the above rules, the resulting canonical request string in the example is as follows (for clarity, \nnewlines are replaced by a new line printed separately):

```
GET
```

```
/
Limit=10&Offset=0
content-type:application/x-www-form-urlencoded
host:cvm.convergecloud.com

content-type;host
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

2.2. Concatenate the string to be signed

Concatenate the string to be signed in the following format:

```
StringToSign =
  Algorithm + \n +
  RequestTimestamp + \n +
  CredentialScope + \n +
  HashedCanonicalRequest
```

- Algorithm: Signature algorithm, currently fixed as TC3-HMAC-SHA256;
- RequestTimestamp: Request timestamp, i.e. the X-TC-Timestamp value of the request header. The request in the example above is 1539084154;
- CredentialScope: Credential range in the format Date/service/tc3_request, containing date, requested service and termination string (tc3_request). **Date is the date of UTC standard time, the value needs to be consistent with UTC standard time converted by common parameter X-TC-Timestamp **;service is the product name, which must be consistent with the product domain name called, such as cvm. The above example request, the value is 2018-10-09/cvm/tc3_request;
- HashedCanonicalRequest: The hash value of the canonical request string obtained by concatenating the above steps is calculated by Lowercase(HexEncode(Hash.SHA256(CanonicalRequest))).

note:

> 1. Date must be calculated from timestamp X-TC-Timestamp and time zone UTC+0. Adding local time zone information to the system, such as East Zone 8, will cause calls to succeed during the day and night, but calls to fail in the early hours of the morning. Assuming the timestamp is 1551113065, the time in the East Eight District is 2019-02-26 00:44:25, but the calculated Date with UTC+0 should be 2019-02-25 instead of 2019-02-26.
> 2. Timestamp must be the current system time and ensure that the system time and standard time are synchronized. If the difference exceeds five minutes, it will fail. If it is not synchronized with the standard time for a long time, it may cause the request to fail after running for a period of time (return signature expiration error).

According to the above rules, the string to be signed in the example is as follows (for clarity,\nnewlines are

replaced by new lines printed):

```
TC3-HMAC-SHA256
1539084154
2018-10-09/cvm/tc3_request
91c9c192c14460df6c1ffc69e34e6c5e90708de2a6d282cccf957dbf1aa7f3a7
```

2.3. calculate the signature

1. Compute the derived signature key, pseudocode as follows

```
SecretKey = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"
SecretDate = HMAC_SHA256("TC3" + SecretKey, Date)
SecretService = HMAC_SHA256(SecretDate, Service)
SecretSigning = HMAC_SHA256(SecretService, "tc3_request")
```

- SecretKey: Original SecretKey;
- Date: That is, the Date field information in Credential, as shown in the example above, is 2018-10-09.;
- Service: That is, the Service field information in Credential, as shown in the example above, is cvm;

2. Calculate the signature, pseudocode as follows

```
Signature = HexEncode(HMAC_SHA256(SecretSigning, StringToSign))
```

- SecretSigning: That is, the derivative signature key calculated above;
- StringToSign: That is, the character string to be signed calculated in step 2;

2.4. Splicing Authorization

Splice as follows Authorization:

```
Authorization =
    Algorithm + ' ' +
    'Credential=' + SecretId + '/' + CredentialScope + ', ' +
    'SignedHeaders=' + SignedHeaders + ', '
    'Signature=' + Signature
```

- Algorithm: Signature method, fixed as TC3-HMAC-SHA256;
- SecretId: SecretId in key pair;
- CredentialScope: See above, Scope of Voucher;
- SignedHeaders: See above, header information for participating signatures;
- Signature: signature value

According to the above rules, the values obtained in the example are:

```
TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
```

The final complete invocation information is as follows:

```
https://cvm.convergecloud.com/?Limit=10&Offset=0

Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE/2018-10-09/cvm/tc3_request, SignedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
Content-Type: application/x-www-form-urlencoded
Host: cvm.convergecloud.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1539084154
X-TC-Region: luzon
```

3. signing failed

According to the actual situation, there are the following signature failure error codes, please handle according to the actual situation

error code	error description
AuthFailure.SignatureExpire	signature expiration
AuthFailure.SecretIdNotFound	key is not present
AuthFailure.SignatureFailure	signature error
AuthFailure.TokenFailure	token error
AuthFailure.InvalidSecretId	Illegal key (not cloud API key type)

4. Signature Demo

Java

```
import java.io.BufferedReader;
import java.io.InputStream;
```

```
import java.io.InputStreamReader;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Map;
import java.util.TimeZone;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.net.ssl.HttpsURLConnection;
import javax.xml.bind.DatatypeConverter;

import org.apache.commons.codec.digest.DigestUtils;

public class CloudAPITC3Demo {
    private final static String CHARSET = "UTF-8";
    private final static String ENDPOINT = "cvm.convergecloud.com";
    private final static String PATH = "/";
    private final static String SECRET_ID = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE";
    private final static String SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE";
    private final static String CT_X_WWW_FORM_URLENCODED = "application/x-www-form-urlencode
d";
    private final static String CT_JSON = "application/json";
    private final static String CT_FORM_DATA = "multipart/form-data";

    public static byte[] sign256(byte[] key, String msg) throws Exception {
        Mac mac = Mac.getInstance("HmacSHA256");
        SecretKeySpec secretKeySpec = new SecretKeySpec(key, mac.getAlgorithm());
        mac.init(secretKeySpec);
        return mac.doFinal(msg.getBytes(CHARSET));
    }

    public static void main(String[] args) throws Exception {
        String service = "cvm";
        String host = "cvm.convergecloud.com";
        String region = "luzon";
        String action = "DescribeInstances";
        String version = "2017-03-12";
        String algorithm = "TC3-HMAC-SHA256";
        String timestamp = "1539084154";
        //String timestamp = String.valueOf(System.currentTimeMillis() / 1000);
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        // Pay attention to the time zone, otherwise it is easy to make mistakes.
        sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
        String date = sdf.format(new Date(Long.valueOf(timestamp + "000")));

        // ************* Step 1: Assemble the specification request string *************
```

```
        String httpRequestMethod = "GET";
        String canonicalUri = "/";
        String canonicalQueryString = "Limit=10&Offset=0";
        String canonicalHeaders = "content-type:application/x-www-form-urlencoded\n" + "host:" + host
+ "\n";
        String signedHeaders = "content-type;host";
        String hashedRequestPayload = DigestUtils.sha256Hex("");
        String canonicalRequest = httpRequestMethod + "\n" + canonicalUri + "\n" + canonicalQueryStri
ng + "\n"
                + canonicalHeaders + "\n" + signedHeaders + "\n" + hashedRequestPayload;
        System.out.println(canonicalRequest);

        // ************* Step 2: Stitching the string to be signed *************
        String credentialScope = date + "/" + service + "/" + "tc3_request";
        String hashedCanonicalRequest = DigestUtils.sha256Hex(canonicalRequest.getBytes(CHARSET));
        String stringToSign = algorithm + "\n" + timestamp + "\n" + credentialScope + "\n" + hashedCan
onicalRequest;
        System.out.println(stringToSign);

        // ************* Step 3: Calculate the signature *************
        byte[] secretDate = sign256(("TC3" + SECRET_KEY).getBytes(CHARSET), date);
        byte[] secretService = sign256(secretDate, service);
        byte[] secretSigning = sign256(secretService, "tc3_request");
        String signature = DatatypeConverter.printHexBinary(sign256(secretSigning, stringToSign)).toLow
erCase();
        System.out.println(signature);

        // ************* Step 4: Splicing Authorization *************
        String authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + credentialScope + ", "
                + "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
        System.out.println(authorization);

        TreeMap<String, String> headers = new TreeMap<String, String>();
        headers.put("Authorization", authorization);
        headers.put("Host", host);
        headers.put("Content-Type", CT_X_WWW_FORM_URLENCODED);
        headers.put("X-TC-Action", action);
        headers.put("X-TC-Timestamp", timestamp);
        headers.put("X-TC-Version", version);
        headers.put("X-TC-Region", region);
    }
}
```

Python

```
# -*- coding: utf-8 -*-
```

```
import hashlib, hmac, json, os, sys, time
from datetime import datetime

# key parameters
secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"

service = "cvm"
host = "cvm.convergecloud.com"
endpoint = "https://" + host
region = "luzon"
action = "DescribeInstances"
version = "2017-03-12"
algorithm = "TC3-HMAC-SHA256"
timestamp = 1539084154
date = datetime.utcfromtimestamp(timestamp).strftime("%Y-%m-%d")
params = {"Limit": 10, "Offset": 0}

# ************* Step 1: Assemble the specification request string *************
http_request_method = "GET"
canonical_uri = "/"
canonical_querystring = "Limit=10&Offset=0"
ct = "x-www-form-urlencoded"
payload = ""
if http_request_method == "POST":
    canonical_querystring = ""
    ct = "json"
    payload = json.dumps(params)
canonical_headers = "content-type:application/%s\nhost:%s\n" % (ct, host)
signed_headers = "content-type;host"
hashed_request_payload = hashlib.sha256(payload.encode("utf-8")).hexdigest()
canonical_request = (http_request_method + "\n" +
                     canonical_uri + "\n" +
                     canonical_querystring + "\n" +
                     canonical_headers + "\n" +
                     signed_headers + "\n" +
                     hashed_request_payload)
print(canonical_request)

# ************* Step 2: Stitching the string to be signed *************
credential_scope = date + "/" + service + "/" + "tc3_request"
hashed_canonical_request = hashlib.sha256(canonical_request.encode("utf-8")).hexdigest()
string_to_sign = (algorithm + "\n" +
                  str(timestamp) + "\n" +
                  credential_scope + "\n" +
                  hashed_canonical_request)
print(string_to_sign)
```

```
# ************* Step 3: Calculate the signature *************
# Compute signature digest function
def sign(key, msg):
    return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()
secret_date = sign(("TC3" + secret_key).encode("utf-8"), date)
secret_service = sign(secret_date, service)
secret_signing = sign(secret_service, "tc3_request")
signature = hmac.new(secret_signing, string_to_sign.encode("utf-8"), hashlib.sha256).hexdigest()
print(signature)

# ************* Step 4: Splicing Authorization *************
authorization = (algorithm + " " +
                 "Credential=" + secret_id + "/" + credential_scope + ", " +
                 "SignedHeaders=" + signed_headers + ", " +
                 "Signature=" + signature)
print(authorization)

# Common parameters added to request header
headers = {
    "Authorization": authorization,
    "Host": host,
    "Content-Type": "application/%s" % ct,
    "X-TC-Action": action,
    "X-TC-Timestamp": str(timestamp),
    "X-TC-Version": version,
    "X-TC-Region": region,
}
```

request

1. Service Address

Region refers to the geographical area of a physical data center. Converge Cloud ensures complete isolation between different regions, providing maximum stability and fault tolerance. To reduce access latency and improve download speed, it is recommended to choose the region closest to your customers.

You can view the complete region list by using the API interface "DescribeRegions".

2. Communications Protocol

All the Converge Cloud APIs communicate via HTTPS, providing highly secure communication tunnels.

3. Request Methods

Supported HTTP request methods:

- POST (recommended)
- GET

PThe Content-Type types supported by POST requests:

- application/json (recommended). The TC3-HMAC-SHA256 signature algorithm must be used.
- application/x-www-form-urlencoded. The HmacSHA1 or HmacSHA256 signature algorithm must be used.
- multipart/form-data (only supported by certain APIs). You must use TC3-HMAC-SHA256 to calculate the signature.

The size of a GET request packet is up to 32 KB. The size of a POST request is up to 1 MB when the HmacSHA1 or HmacSHA256 signature algorithm is used, and up to 10 MB when TC3-HMAC-SHA256 is used.

4. Character Encoding

Only UTF-8 encoding is used.

result

Response for Successful Requests

For example, when calling CAM API (version: 2017-03-12) to view the status of instances (DescribeInstancesStatus), if the request has succeeded, you may see the response as shown below:


```
{
    "Response": {
        "TotalCount": 0,
        "InstanceStatusSet": [],
        "RequestId": "b5b41468-520d-4192-b42f-595cc34b6c1c"
    }
}
```


- The API will return Response, which contains RequestId, as long as it processes the request. It does not matter if the request is successful or not.
- RequestId is the unique ID of an API request. Contact us with this ID when an exception occurs.
- Except for the fixed fields, all fields are action-specified. For the definitions of action-specified fields, see the corresponding API documentation. In this example, TotalCount and InstanceStatusSet are the fields specified by the API DescribeInstancesStatus. 0 TotalCount means that the requester owns 0 CVM instance so the InstanceStatusSet is empty.

Response for Failed Requests

If the request has failed, you may see the response as shown below:


```
{
    "Response": {
        "Error": {
            "Code": "AuthFailure.SignatureFailure",
            "Message": "The provided credentials could not be validated. Please check your signature is cor
rect."
        },
        "RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
    }
}
```


- The presence of the Error field indicates that the request has failed. A response for a failed request will include Error, Code and Message fields.

- Code is the code of the error that helps you identify the cause and solution. There are two types of error codes so you may find the code in either common error codes or API-specified error codes.
- Message explains the cause of the error. Note that the returned messages are subject to service updates. The information the messages provide may not be up-to-date and should not be the only source of reference.
- RequestId is the unique ID of an API request. Contact us with this ID when an exception occurs.

Common Error Codes

If there is an Error field in the response, it means that the API call failed. The Code field in Error indicates the error code. The following table lists the common error codes that all actions can return.

Error Code	Description
AuthFailure.InvalidSecretId	Invalid key (not a API key type).
AuthFailure.MFAFailure	MFA failed.
AuthFailure.SecretIdNotFound	The key does not exist.
AuthFailure.SignatureExpire	Signature expired.
AuthFailure.SignatureFailure	Signature error.
AuthFailure.TokenFailure	Token error.
AuthFailure.UnauthorizedOperation	The request does not have CAM authorization.
DryRunOperation	DryRun Operation. It means that the request would have succeeded, but the DryRun parameter was used.
FailedOperation	Operation failed.
InternalError	Internal error.
InvalidAction	The API does not exist.
InvalidParameter	Incorrect parameter.
InvalidParameterValue	Invalid parameter value.
LimitExceeded	Quota limit exceeded.
MissingParameter	A parameter is missing.
NoSuchVersion	The API version does not exist.

Error Code	Description
RequestLimitExceeded	The number of requests exceeds the frequency limit.
ResourceInUse	Resource is in use.
ResourceInsufficient	Insufficient resource.
ResourceNotFound	The resource does not exist.
ResourceUnavailable	Resource is unavailable.
UnauthorizedOperation	Unauthorized operation.
UnknownParameter	Unknown parameter.
UnsupportedOperation	Unsupported operation.
UnsupportedProtocol	hHTTPS request method error. Only GET and POST requests are supported.
UnsupportedRegion	API does not support the requested region.

PublicParameter

Common parameters are parameters used to identify users and interface authentication purposes. If not necessary, these parameters are not described in the separate interface document of each interface, but these parameters need to be carried in each request before the request can be normally initiated.

signature method v3

When using TC3-HMAC-SHA256 signature method, the common parameters need to be uniformly placed in the HTTP Header request header, as follows:

parameter name	type	required	described
X-TC-Action	String	Yes	The name of the operation interface. Refer to the description of the public parameter Action in the input parameters of the interface documentation for the value. For example, the value for the query instance list interface of the cloud server is DescribeInstances.
X-TC-Region	String	Yes	region parameters, Data used to identify which region you wish to operate on. The region value accepted by the interface refers to the description of the input parameter common parameter Region in the interface document. Note: Some interfaces do not need to pass this parameter, which is specifically stated in the interface documentation. In this case, even passing this parameter will not take effect.
X-TC-Timestamp	Integer	Yes	The current UNIX timestamp, which records the time when the API request was initiated. for example 1529223702. Note: If the time difference from the server exceeds 5 minutes, it will cause a signature expiration error.
X-TC-Version	String	Yes	Version of API for operation. Refer to the description of the public parameter Version in the interface document for the value. For example, the version of the cloud server. 2017-03-12.
Authorization	String	Yes	HTTP The standard authentication header field is "Authorization". For example: TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024 Here, - TC3-HMAC-SHA256: Signature method, currently fixed as this value, - Credential: Credential: Signature credential; AKIDEXAMPLE is the SecretId; Date is a date in UTC time, and this value must match the value of X-TC-Timestamp (a common parameter) in UTC time format; service is the name of the product/service, and is generally a domain name prefix. For example,cvm; - SignedHeaders: The headers that contains the authentication information; content-type and host are the required headers; - Signature: Signature digest.
X-TC-Token	String	No	The token used for temporary certificates needs to be used in conjunction with a temporary key. Temporary keys and tokens need to be obtained by calling the access management service interface. Long-term keys do not require tokens.

signature method v1

When using HmacSHA1 and HmacSHA256 signature methods, common parameters need to be placed uniformly in the request string, as follows

parameter name	type	required	described
Action	String	Yes	The name of the interface to be operated. The value refers to the explanation of the public parameter Action in the interface documentation. For example, for the cloud server's query instance list interface, the value is DescribeInstances .
Region	String	Yes	The region parameter is used to identify which region's data you want to operate on. The region value accepted by the interface is specified in the input parameter of the public parameter Region in the interface documentation. Note: Some interfaces do not require this parameter to be passed, which will be specifically mentioned in the interface documentation. Even if this parameter is passed, it will not take effect in this case.
Timestamp	Integer	Yes	The current UNIX timestamp, which can record the time when the API request is initiated. For example, 1529223702. If the difference between the current time is too large, it will cause a signature expiration error.
Nonce	Integer	Yes	A random positive integer combined with Timestamp is used to prevent replay attacks.
SecretId	String	Yes	The SecretId is an identifier applied for the cloud API key to identify the user's identity. A SecretId corresponds to a unique SecretKey , and the SecretKey is used to generate the request signature Signature .
Signature	String	Yes	Request signature is used to verify the legitimacy of this request. Users need to calculate it based on the actual input parameters. The specific calculation method can be found in the interface authentication documentation.
Version	String	Yes	The version of the API being operated on. The value refers to the description of the public parameter Version in the interface documentation. For example, the version of the cloud server. 2017-03-12.
SignatureMethod	String	No	Signature method, currently supports HmacSHA256 and HmacSHA1. HmacSHA256 algorithm is used to verify signatures only if this parameter is specified as HmacSHA 256, otherwise HmacSHA1 is used to verify signatures.
Token	String	No	Token used for temporary certificate needs to be used together with temporary key. The temporary key and Token need to be obtained from the Access Management service invocation interface. Long-term keys do not require tokens.

zone list

A region (Region) refers to the geographical area of a physical data center. Converge Cloud Translation: To deliver verification that completely isolates different regions, ensuring maximum stability and fault tolerance between regions. In order to reduce access latency and improve download speed, it is recommended that you choose the region closest to your customers.

You can view the complete region list by using the API interface Query Region List.

Cross-Cloud Replication CheckBucketCrossCloudReplicationDestination

1. Interface Description

Interface request domain name: cos.api3.convergecloud.com.

This API is used to check the connectivity of the target bucket for cross-cloud replication.

Default interface request rate limit: 20 times per second.

Interface update time: 2024-05-22 15:10:41.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: CheckBucketCrossCloudReplicationDestination
Version	Yes	No	String	Public parameter, The value of this interface: 2020-01-07
Region	No	No	String	Public parameter,Regional information This interface does not need to pass this parameter.
Params	Yes	No	CrossCloudReplicationDestinationConfiguration	Target bucket for cross-cloud replication. Example:View

3. Output Parameters

Parameter name	Type	Description
Code	String	Check result code. "ok" indicates success, and any other value indicates failure. The failure reason can be found in the Message field. Example:
Message	String	Reason for check failure. Adapt language according to the Language parameter in the request. Example:
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

There is no error code related to business logic for this interface, For other error codes, see public ErrorCode.

GetBucketCrossCloudReplication

1. Interface Description

Interface request domain name: cos.api3.convergecloud.com.

This API is used to obtain cross-cloud replication configuration of a bucket.

Default interface request rate limit: 20 times per second.

Interface update time: 2024-05-22 15:11:03.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: GetBucketCrossCloudReplication
Version	Yes	No	String	Public parameter, The value of this interface: 2020-01-07
Region	No	No	String	Public parameter,Regional information This interface does not need to pass this parameter.
Params	Yes	No	GetBucketCrossCloudReplicationConfiguration	Cross-cloud replication parameters. Example:View

3. Output Parameters

Parameter name	Type	Description
Code	String	API response status. "ok" indicates success, and any other value indicates failure. The failure reason can be found in the Message field. Example:
Message	String	Reason for API failure. Adapt language according to the Language parameter in the request. Example:
Data	Array of CrossCloudReplicationConfiguration	Disaster recovery configuration list. Example:View
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

There is no error code related to business logic for this interface, For other error codes, see public ErrorCode.

PutBucketCrossCloudReplication

1. Interface Description

Interface request domain name: cos.api3.convergecloud.com.

This API is used to configure cross-cloud replication for a bucket.

Default interface request rate limit: 20 times per second.

Interface update time: 2025-05-06 14:51:28.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: PutBucketCrossCloudReplication
Version	Yes	No	String	Public parameter, The value of this interface: 2020-01-07
Region	No	No	String	Public parameter,Regional information This interface does not need to pass this parameter.
Params	Yes	No	PutBucketCrossCloudReplicationConfiguration	Cross-cloud replication parameters. Example:View

3. Output Parameters

Parameter name	Type	Description
Code	String	API response status. "ok" indicates success, and any other value indicates failure. The failure reason can be found in the Message field. Example: ""
Message	String	Reason for API failure. Adapt language according to the Language parameter in the request. Example: ""
Data	Array of CrossCloudReplicationConfiguration	Disaster recovery configuration list. Example:View
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

There is no error code related to business logic for this interface, For other error codes, see public ErrorCode.

Data structure

ListAccessPathResp

ListAccessPathResp

Referenced by the following interfaces: ListAccessPath

name	required	NULL allowed	type	description
Id	yes	no	Int64	Id Example: 25
AccessPath	yes	no	String	AccessPath Example: examplebucket-1255000034/path1
CosRegion	yes	no	String	CosRegion Example: region1

CrossCloudReplicationConfiguration

Cross-cloud replication configuration.

Referenced by the following interfaces: GetBucketCrossCloudReplication, PutBucketCrossCloudReplication

name	required	NULL allowed	type	description
Status	yes	no	String	Configuration status (Enabled
Destination	yes	no	CrossCloudReplicationDestinationConfiguration	Target bucket for cross-cloud replication. Example:View
SourcePrefix	no	no	String	Prefix of the source bucket for cross-cloud replication. Example: ""
DestinationPrefix	no	no	String	Prefix of the target bucket for cross-cloud replication. Example: ""
PrefixType	no	no	Bool	Allowlist or blocklist type. The default value is false, indicating the allowlist. Example: false
Concurrency	no	no	Int64	Number of replication concurrencies. Example: 0
RequestLimit	no	no	Int64	Frequency control configuration. Example: 0
TrafficLimit	no	no	Int64	Flow control configuration. Example: 0

stopService

stopService

Referenced by the following interfaces: GetUserStat

name	required	NULL allowed	type	description
Date	no	no	String	date Example: ""
IsStopService	no	no	Bool	isStopService Example: false
Reason	no	no	String	reason Example: ""
ReasonType	no	no	String	reason_type Example: ""
Status	no	no	Int64	status Example: 0

GetBucketCrossCloudReplicationConfiguration

Request parameters for obtaining cross-cloud replication configuration of a bucket.

Referenced by the following interfaces: GetBucketCrossCloudReplication

name	required	NULL allowed	type	description
Region	yes	no	String	Region of a COS bucket. Example:
Bucket	yes	no	String	COS bucket name. Example: bucket1-1255000117

GetBillingTypeData

Data structure in the output parameters of GetBillingType.

Referenced by the following interfaces: GetBillingType

name	required	NULL allowed	type	description
IsBilling	no	no	Bool	Is the current COS deployment billed? Example: false

StdGet404num

std_get_404num

Referenced by the following interfaces: GetStatHttpDay

name	required	NULL allowed	type	description
StdGet404num	no	no	Array of Int64	StdGet404num Example: 0

PutBucketCrossCloudReplicationConfiguration

Request parameters for configuring cross-cloud replication for a bucket.

Referenced by the following interfaces: PutBucketCrossCloudReplication

name	required	NULL allowed	type	description
Region	yes	no	String	Region of a source bucket. Example:
Bucket	yes	no	String	Source bucket name. Example: bucket1-1255000117
Configuration	yes	no	Array of CrossCloudReplicationConfiguration	Cross-cloud replication configuration list. Example:View

Bucket

Bucket Information

Referenced by the following interfaces: ListBuckets

name	required	NULL allowed	type	description
Name	yes	no	String	Bucket Name Example: cos-bucket1-1255000034
Location	yes	no	String	COS Region of Bucket Example: region1
CreationDate	yes	no	String	Bucket Creation Time Example: 2025-03-11T02:57:34Z

QuotaBucketDefault

QuotaBucketDefault

Referenced by the following interfaces: DescribeBucketDefaultQuota

name	required	NULL allowed	type	description
MaxCapacity	no	no	String	MaxCapacity Example: unlimit
MaxObjects	no	no	String	MaxObjects Example: unlimit
TouchTime	no	no	String	TouchTime Example: 2025-03-14T17:38:43Z

CrossCloudReplicationDestinationConfiguration

Target bucket configuration for cross-cloud replication.

Referenced by the following interfaces: CheckBucketCrossCloudReplicationDestination, GetBucketCrossCloudReplication, PutBucketCrossCloudReplication

name	required	NULL allowed	type	description
Endpoint	yes	no	String	S3 bucket endpoint. Example:
Bucket	yes	no	String	S3 bucket name. Example: bucket1-1255000117
SecretID	yes	no	String	S3 bucket access key ID. Example:
SecretKey	yes	no	String	S3 bucket access key. Example:
SecretToken	no	no	String	S3 bucket access key token. Example:

StatDayData

StatDayData

Referenced by the following interfaces: GetStatDay, GetStatDays, GetStatHttpDays

name	required	NULL allowed	type	description
file_count	no	no	Array of Int64	file_count Example: 1
file_count_si	no	no	Array of Int64	file_count_si Example: 0
get_count	no	no	Array of Int64	get_count Example: 1
get_count_si	no	no	Array of Int64	get_count_si Example: 0
idc_traffic	no	no	Array of Int64	idc_traffic Example: 12
internel_traffic	no	no	Array of Int64	internel_traffic Example: 11
idc_traffic_si	no	no	Array of Int64	idc_traffic_si Example: 0
internel_traffic_si	no	no	Array of Int64	internel_traffic_si Example: 0
put_count	no	no	Array of Int64	put_count Example: 2
put_count_si	no	no	Array of Int64	put_count_si Example: 0

name	required	NULL allowed	type	description
storage	no	no	Array of Int64	storage Example: 88
storage_si	no	no	Array of Int64	storage_si Example: 0

ErrorCode

Feature Description

If there is an Error field in the response, it means that the API call failed. For example:

```
{
    "Response": {
        "Error": {
            "Code": "AuthFailure.SignatureFailure",
            "Message": "The provided credentials could not be validated. Please check your signature is cor
rect."
        },
        "RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
    }
}
```

Code in Error indicates the error code, and Message indicates the specific information of the error.

Error Code List

Common Error Codes

Error codes	Description
AuthFailure.InvalidSecretId	Invalid key (not a Cloud API key type).
AuthFailure.MFAFailure	MFA failed.
AuthFailure.SecretIdNotFound	Key does not exist. Check if the key has been deleted or disabled in the console, and if not, check if the key is correctly entered. Note that whitespaces should not exist before or after the key.
AuthFailure.SignatureExpire	Signature expired. Timestamp and server time cannot differ by more than five minutes. Please ensure your current local time matches the standard time.
AuthFailure.SignatureFailure	Invalid signature. Signature calculation error. Please ensure you've followed the signature calculation process described in the Signature API documentation.
AuthFailure.TokenFailure	Token error.

Error codes	Description
AuthFailure.UnauthorizedOperation	The request does not have CAM authorization.
DryRunOperation	DryRun Operation. It means that the request would have succeeded, but the DryRun parameter was used.
FailedOperation	Operation failed.
InternalError	Internal error.
InvalidAction	The API does not exist.
InvalidParameter	Incorrect parameter.
InvalidParameterValue	Invalid parameter value.
LimitExceeded	Quota limit exceeded.
MissingParameter	A parameter is missing.
NoSuchVersion	The API version does not exist.
RequestLimitExceeded	The number of requests exceeds the frequency limit.
ResourceInUse	Resource is in use.
ResourceInsufficient	Insufficient resource.
ResourceNotFound	The resource does not exist.
ResourceUnavailable	Resource is unavailable.
UnauthorizedOperation	Unauthorized operation.
UnknownParameter	Unknown parameter.
UnsupportedOperation	Unsupported operation.
UnsupportedProtocol	HTTPS request method error. Only GET and POST requests are supported.
UnsupportedRegion	API does not support the requested region.

Service Error Codes