Table of Contents

Overview

We provide a set of RESTful APIs compatible with AmazonS3, which can give you more freedom to develop flexible features.

BaishanCloud Storage Service provides the following three main types of APIs.

- Service operations
- Bucket operations
- Object operation

At the same time, to improve the security of user usage, White Mountain Cloud Storage Service also verifies the identity of the requestor by using signatures.

For more information about the signature algorithm, please refer to Signature Algorithm.

Note: The examples used in the following interfaces are in the context of requiring the use of signatures; if the associated access resource has been set to be accessible anonymously (to all users), it may not carry a signature.

白山云存储服务(BaishanCloud storage service)

BaishanCloud storage service Intro

BaishanCloud Storage Service: It is a multi-data center, multi-copy, distributed architecture, object cloud storage service extended based on the advantages of BaishanCloud's powerful CDN-X service.

Functionality Overview：

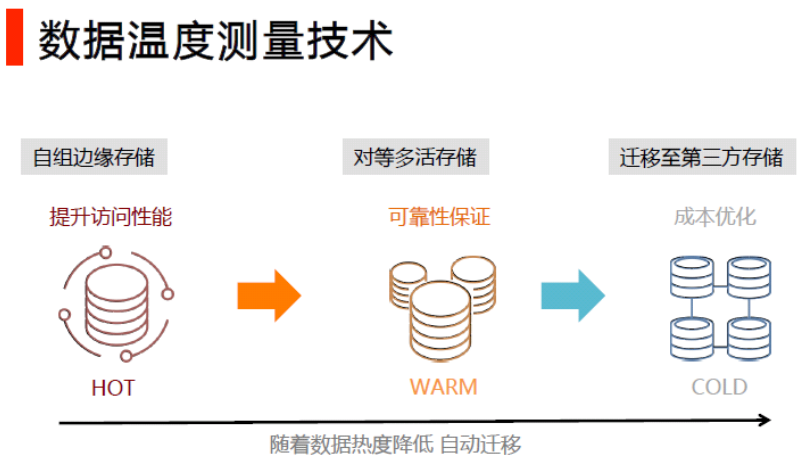
1. High reliability and high performance: through multi-data center, multi-copy, distributed architecture, multi-device redundancy, cross-regional replication, off-site disaster recovery, internal intelligent scheduling of the system to ensure stable, durable and reliable for user data storage; through the CDN-X edge nodes for accelerated upload, through the whole network CDN for accelerated download.
2. High security and protection: There are multiple security, reliability and user resource isolation mechanisms inside the system; precise data authority management, multi-level decentralization, multiple authentication and multiple authorization management mechanisms to prevent hotlinking and protect data security.
3. Flexible expansion, focus on service: unlimited expansion, no migration, to meet the growth trend of unstructured data, distributed and analytical functions; configuration of exclusive technical service interface people to understand user needs in depth, follow up user services.
4. Fully functional, easy to use: the core members of the R&D team have accumulated more than 7 years of experience in object cloud storage services, all independent intellectual property rights, self-developed core components, high concurrency production environment verification; user programming interface is simple, easy to use, stable and reliable

Technical advantages：

1. Powerful CDN advantage: Accelerate uploading through BaishanCloud's CDN-X edge node, accelerate downloading through the whole BaishanCloud CDN network.
2. Follow compatible specifications: follow and compatible with Amazon S3 REST API interface, recommend users directly use the official SDK from Amazon S3, simple and easy to use, stable and reliable, users can use a variety of open-source tools, etc.
3. Audio and video transcoding and image processing: provide offline audio and video transcoding, and easy image processing interface (format conversion, rotation, scaling, cropping, compression, watermark, blur, and other filter operations), users can efficiently process static images online to avoid repeated upload/download images.
4. Powerful object cloud storage service architecture design: realize data temperature measurement, self-group edge storage, peer-to-peer multi-live, and self-activated quality control.

4 Technical innovations:

1. Data temperature measurement

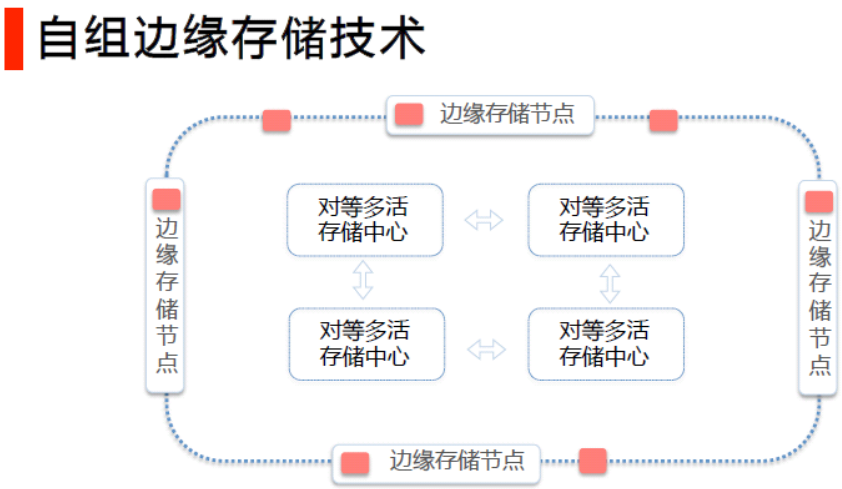


1. Define data: Define data details in a total of six dimensions: data size, type, source, access frequency, and trustworthiness. Internal mechanism will schedule data grading based on access frequency. Hot data is automatically migrated and stored with high I/O.
2. Tiered data storage: After tiered hot and cold data, the hot data is provided with higher I/O by 3-copy strategy, while the cold data is provided with higher reliability. Through the data storage statistics, the data that carries 90% of the

system's I/O is defined as hot data and the rest is defined as cold data, and the 3-copy of hot data is defined as cold data within the system periodically.

3. Asynchronous periodic merging: The hot/cold separation uses a hierarchical storage method, setting the newer data and the older data into different levels, and periodically merging the newer data from the upper level and the older data from the lower level into the lower level. This asynchronous periodic merging reduces the I/O updates consumption of cold data by 1 to 2 orders of magnitude.

2. Self-organized edge nodes



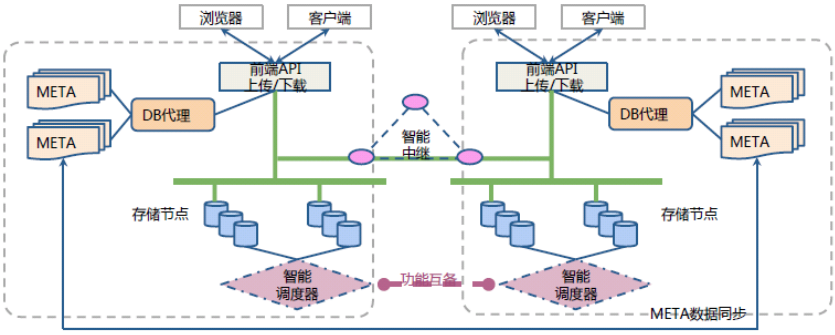
1. Unified network planning: Through p2p broadcast network, our edge nodes are connected into one network to notify user’s file uploads to every server in the country within 1000ms, to achieve the ultimate consistency of user data with low latency. It allows all remote users to quickly access to the files uploaded.

2. Fiber optic link access: Edge nodes maintain communication with 5 central storage server rooms at the same time, and asynchronously transfer user uploaded files to the central storage within seconds through the data transfer acceleration network, ensuring the reliability of data up to 99.99999999999%.

3. Isolation of network failure: When the central storage fails or the edge storage nodes cannot communicate with the central storage, the edge nodes can also support users' upload and download requests within a certain period to isolate network failure to users.

3.Multi-layer high redundancy

对等多活存储技术



1. High-quality multi-data center interconnection: multiple central storage fiber interconnection, synchronization of large files in seconds. Provide database offsite multiple writes and reads: completely circumvent the service availability problem caused by server room failure. By deploying multiple primary and backup copies in multiple server rooms, the database can be read and written simultaneously in multiple locations to complete off-site data backup function.
2. Cross-server synchronization in seconds: Ensure that when any server fails, it will not affect the service, truly achieving network-wide redundancy and disaster recovery across multiple server rooms.
3. Outage response in seconds: The failure discovery mechanism provides fast response speed in seconds. Automatically isolate unavailable database services, change read and write functions to available service nodes.
4. Database detection in seconds: Detect the database instance availability every second, and when it is not connected, immediately switch the database and update the configuration information. Each database agent monitors the configuration changes in real time and reloads and takes effect in 10 milliseconds when the configuration changes. To achieve high availability of data connections.
5. Configuration decentralization: The backup configuration is deployed to multiple server rooms, so that when one server room fails, the system can still work normally for configuration changes and failure discovery.

4. Self-excitation quality control

1. paxos algorithm and automated data repair: paxos algorithm is used to automate the management of multiple copies, data self-repair, and consistency between multiple copies, etc. The entire storage system cluster is divided into 100,000 self-maintaining, self-repair groups, each group regularly detects the status of other nodes, data accuracy, and data repair at the granularity of minutes.
2. Improvements to paxos: First, the paxos algorithm guarantees the accuracy of automated data repair. paxos itself does not contain a description of the algorithm for paxos group management. Online environments using the paxos algorithm must also consider the handling of group member changes. The BaishanCloud Storage System introduces a self-designed and implemented paxos group management algorithm that enables group member changes through the operation of two phases of paxos instances. During the change process, the member change algorithm ensures data consistency, and the change result will not be affected by network or disk failure.
3. Ensure the security of operation process: Any network or disk failure during the process of member change will not cause the system error, and the service can continue to run after restart.
4. Comprehensive quality testing: 60 seconds to switch write points according to disk status; 5 minutes to check all copies status; 24 hours to complete copies data comparison and self-repair; 1 month for full data scanning and error correction.

Scale and scalability of services:

Service items	Scalable numbers
Number of users the service can carry	1 million users
Service data storage capacity	100+ PB level, 100+ billion files
Data write capacity (upload)	500+ TB in 24 hours (currently 40TB)
Overall service throughput (download)	10+ PB 24-hour downloads (currently 1PB)
Number of physical servers	3000 (multiple data centers)

Service-Level Agreement (SLA).

Reliability: 99.9999999999999%.

Average upload/download speed by ISPs for Internet backbone nodes.

Transfer file size	Upload speed	Download speed
50KB	300KB/s	900KB/s
2MB	750KB/s	2MB/s
4MB	1MB/s	2.5MB/s

Compliant and compatible with Amazon S3's interface, it is recommended that developers can directly use the official Amazon S3 SDK:

Instruction documentation	SDK package download
Android documentation：AWS Mobile SDK for Android Documentation	SDK download:aws-android-sdk
Browser documentationGetting Started with the SDK in the Browser	SDK download:Bower
IOS documentation：AWS Mobile SDK iOS Developer Guid	SDK download: aws-ios-sdk
Java documentation：AWS SDK for Java Developer Guide	SDK download: aws-java-sdk
.NET documentation：AWS SDK for .NET Documentation	SDK download: AWSToolsAndSDKForNet.msi
Node.js documentation：AWS SDK for JavaScript Documentation	SDK download: Getting Started with the SDK in Node.js
PHP documentation：AWS SDK for PHP Documentation	SDK download: Installing via Composer
Python documentation：Boto 3 Documentation	SDK Download link and description:Boto 3 - The AWS SDK for Python
Ruby documentation：AWS SDK for Ruby	SDK download：AWS SDK for Ruby Developer Guide v2.16.8
Go documentation：AWS SDK for Go	SDK download：aws-sdk-go
C++ documentation：AWS SDK for C++ Documentation	SDK download：aws-sdk-cpp

Advanced technology explanation

1. BaishanCloud Storage Service & BaishanCloud CDN-X

CDN-X acceleration supports:

- CDN upload acceleration
- CDN virtual domain name back to the origin
- CDN token to protect users' data security.

- Third-party CDN back to origin

CDN Edge nodes upload:

BaishanCloud Storage enhances the speed of uploads by working with CDN edge nodes.

Users request an accelerated upload of their domain name, and the DNS server intelligently returns the IP of the nearest CDN edge node to the user.

Users upload files to the nearest edge node, and the edge node transfers data to the storage center through the optimal routing.

By comparing the speed of uploading by edge nodes with directly uploading to the storage center, the average speed is more than doubled, which greatly improves the upload experience.

Configure to use accelerated upload domain to accelerate upload function.

Configuration methods for CDN-X and cloud storage:

To test the download acceleration through CDN, you need to point your domain to the CNAME that CDN provides.

For example CNAME: www.qq.com.i.qingcdn.com, origin is ss.bscstorage.com.

Suggestions for creating a bucket:

1. Create a bucket with the same name as the user's domain name. e.g. if the user's website domain name is www.baishancloud.com, then create a bucket with the name of www.baishancloud.com, then there is no need to modify the host header when the CDN is configured.
2. Create a bucket with user-defined name: for example, the user website domain name is www.baishancloud.com, the user wants to create a bucket named QQ, then on CDN acceleration acceleration, when user requests www.baishancloud.com.s2.i.baishancloud.com, you need to change the host header to qq.ss.bscstorage.com.

404 status and how to pull the file from origin:

If the file does not exist on BaishanCloud Storage, you can configure to go to one specific domain to pull the content.

The file pull is asynchronous, and the completion time is related to the size of the file.

The request also returns a 302, and the value of the location response header is the domain where the file to be pulled.

When using the pull function, you need to provide：

- The domain we use to go back to origin.
- The mapping rule from request uri to origin file uri.

- The token verification method on back to origin request (optional).
- Whether to enable the HEAD request to pull the file from origin.

How to set the caching policy for origin storage:

- BaiShanCloud Storage can configure HTTP request headers such as Cache-Control of the stored files and deploy the caching policy together with HTTP access requests.
- BaishanCloud CDN can enforce the caching policy according to origin on the Cache-Control setting in the HTTP request header.

2. Concept of object storage, directory structure, permission setting, transfer import

Concept of Bucket and Object

- Bucket is the basic container that holds the data. Everything that you store in Cloud Storage such as objects must be contained in a bucket.
- Each piece of data is an object, for instance we can treat each file as an object.
- Users can create buckets, delete buckets, query objects in buckets, and set bucket properties.
- The user can create an object (this object is the file uploaded by the user), delete an object, download the object, set the access rights of the object, etc.
- For example: 1. Create a bucket, the bucket name is: baishan_yun, and the requested url is: PUT /baishan_yun.

Directory structure of object storage:

The object stored in BaishanCloud is not a file in the traditional sense, as well as the storage method of how file stored in the file system. Each Object looks like a file under a directory, but is just a directory-like string for cloud storage. This doesn't affect how the user list files from the specified directory:

When sending a request to list files, set the prefix parameter to the name of the directory prefix, you can only list the files in this directory, for example:

- bucket/1.mp4
- bucket/cc/2.mp4
- bucket/cc/3 .mp4
- bucket/dd/ee/ff/3.mp4

When the requested `prefix` is specified as `cc` , it will list:

- 2.mp4
- 3.mp4

This process is as same as listing files from a directory. In addition to simulating directory listing files, it also provides functions such as specifying the maximum number of files to list, specifying to list files after a certain file name (sorted in alphabetical order), and merging files according to common prefixes. For details, please refer to the API documentation.

Things to pay attention to when using:

- Object name should not start with '/'.

  For example: You need to upload a file called file.txt, bucket is `sandbox`.

  - Object is set to `file.txt`, when accessing it should be `http://ss.bscstorage.com/sandbox/file.txt` (recommended)
  - If object is set to `/file.txt`, it should be `http://ss.bscstorage.com/sandbox//file.txt` when accessing (strongly not recommended)
- It is best not to have two consecutive '/' in the object name. For example: You need to upload a file called file.txt in the folder directory, and the bucket is `sandbox`,

  - Object is set to `folder/file.txt`, when accessing it should be `http://ss.bscstorage.com/sandbox/folder/file.txt` (recommended method)
  - If object is set to `folder//file.txt`, it should be `http://ss.bscstorage.com/sandbox/folder//file.txt` when accessing (strongly not recommended)
  - Why does BaishanCloud Storage not recommend two '//' in the URL: some CDN cache servers may combine two consecutive '//' in the URL into one '/' round-trip source, in this case, the Authentication failed or 404

如何删除白山云存储中的目录：

白山云存储删除目录的方法为： 白山云存储不支持直接删除一个目录, 因为 Amazon S3 API 没有严格意义上的目录的概念(没有目录层级的结构,但习惯上把 Object 按照"/"分隔当做目录来使用),这种情况下,对多个文件操作的原子性方面没法保证，所以这方面都是写脚本实现的。

可以通过以下方式完成该操作：

- 通过List Objects接口列出该目录的所有文件。
- 通过Delete Object接口逐一进行删除。
- 删除目录。

请注意：

- List Objects接口返回的文件列表不全部是该目录下的文件（列表是根据文件名字符排序的）， 可能包含下一个或几个目录；所以需要根据实际情况调整max-keys参数的值， 并且需要对每一个文件进行筛选，确认是该目录的文件;

如果设置 `prefix` 参数的话，就可以严格的列出以该 `prefix` 下所有文件.

- 一次删除（列出所有文件，逐一删除）不能保障该目录的文件都被清理完成 在删除的过程中可能会有新上传，所以整个过程不能保持原子性。

Access control permission details:

the authorized person can be a user, a group, or all users.

1. Control permission for the Bucket:

```
Read: allow the authorized person to list all objects in the bucket.
Writable: Allows the grantee to create, overwrite, and delete objects in the b
Readable ACL: Allows the grantee to read the bucket's ACL.
Writable ACL: Allows the grantee to modify the bucket's ACL.
All permissions: Allows the authorized person to read, write, read ACL, and wr
```

1. Control permission for Object:

```
Readable: Allows the authorized person to read the data and meta information o
Readable ACL: Allows the grantee to read the ACL of the object.
Writable ACL: Allows the grantee to modify the object's ACL.
Full Permissions: Allows the authorized person to read, read ACL, and write AC
```

How to operate buckets and objects without signatures:

Normal operations require signatures to verify user permissions, but it is also allowed to set common attributes to allow users to operate without signatures.

- After the bucket permission is set to public-read, any user can list the files under the bucket, and the request does not require a signature.
- After the bucket permission is set to public-read-write, any user can list the files in the bucket and upload files to the bucket. The request does not require a signature.
- When uploading a file, after specifying the object's permission as public-read, any user can download the file without a signature, it is called anonymous download.
- When a user accesses the storage through the CDN back-to-source, because the CDN is configured with super permissions, the request does not require a signature to access the files on the storage.

4 ways to import data:

1. The user uses the SDK upload file interface to upload the file to the storage.
2. The user uses the SDK to generate the offline download task, and the offline download subsystem of the storage is responsible for downloading the data from the origin site and synchronizing it to the storage.

3. The user configures the 404 back-to-origin rule. When the end-user accesses the bucket, it will automatically go back to the origin site to capture the data and synchronize it to the storage.
4. The user provides the file download list, and BaishanCloud Storage platform downloads and synchronizes the data to storage through the file list.

HTTPS setting up:

1. Before setting up the HTTPS for a domain, the domain must acquire an ICP license. And users can submit the SSL certificate of the domain name to BaishanCloud console, after the uploading is completed and the SSL certificate is associated with the domain, the HTTPS is enabled for the domain.
2. If you use the domain name provided by BaishanCloud Storage, for example: https://ss.bscstorage.com, note: the bucket must be placed in the url request instead of the domain name.

Support for FTP and rsync upload:

BaishanCloud storage service does not directly support the FTP and rsync protocols because of the control of file permissions of the interface.

But users can use the following tools to achieve FTP, rsync upload.

- A graphical FTP alternative can use [cyberduck][cyberduck] for Mac or [s3browser][s3browser] for Windows.
- An alternative to command line FTP or rsync can use [s3cmd][s3cmd]. [s3cmd]: http://s3tools.org/s3cmd[boto]: http://boto.cloudhackers.com/en/latest/[cyberduck]: https://cyberduck.io/[s3browser]: http:/ /s3browser.com

3 steps for large file upload:

1. First call the range upload initialization interface, and the request returns an ‘uploadId’. The next 2 steps require this ‘uploadId’ to identify each range file to be uploaded.
2. Divide the large file into several pieces, call the piece upload interface, upload each piece, and identify the order of the pieces through the ‘partNmuber’ parameter in the request.
3. After the upload is complete, call the range completion interface to notify the storage that the large file upload is complete.

The total data capacity and number of data objects stored in BaishanCloud are not limited. Individual data objects can be between 1 byte and 1 TB in size. The maximum data object that can be uploaded in a single PUT is 1TB. It is recommended to use the range upload interface to upload large files. The specifications for range upload are:

- The maximum size of each slice is 512MB.
- The maximum number of slice is 2000.
- The maximum size of a single file is 1TB.

How to use offline download to synchronize data:

The offline download service supports HTTP protocol, BT protocol, magnet link, etc.

By submitting an offline download task, the offline download service will download the file from the website according to the file address provided in the task and synchronize it to the storage. After the synchronization is completed, the operation result is sent to the user.

Supports various functions such as download task status query, task cancellation, download data corruption check, and fast message transmission within seconds, the performance can reach up to 2000 rps/s .

How to use:

- Using the SDK, send a request to offline.bscstorage.com. The requested url is: http://offline_domain/, fill in the desired bucket name in the section.
- Send the request body as an offline task, for example:

```
{   "Url":"http://www.abc.com/download/movie.mp4",
    "Key":"movies/2016/movie.mp4",
    "SHA1":"abcdeabcdeabcdeabcdeabcdeabcdeabcdeabcde",
    "MD5":"abcdabcdabcdabcdabcdabcdabcdabcd",
    "CRC32":"abcdef",
    "SuccessCallbackUrl":"http://website/succ/",
    "FailureCallbackUrl":"http://website/fail/",
}
```

- The user needs to start a service to receive the offline download results sent by the offline download service.

3. Audio and video transcoding, static image processing, self-service management console

Audio and video transcoding: support transcoding between formats such as flv, mp4, mov, etc.

Support reducing bit rate, resolution, modifying gop (key frame interval), video screenshots and other functions.

BaishanCloud cloud storage static file processing:

- Supports format conversion, rotation, scaling, cropping, image-text mixing, watermarking and custom image processing of static images, see imgx_manual documentation for details.
- After the requested image is processed and cached, the next request will not enable the processing procedure repeatedly. The default coaching time is 7 days.
- If a CDN service is configured and enabled, the processed images will be automatically pushed to the CDN edge nodes.
- If the original image is modified, you can use the v command to regenerate the image and link.

Account and testing environment creation:

Please provide enterprise information and contact information for each account, the account team will manually audit the provided information.

- BaishanCloud console creates a user, generates the accseskey and secretkey corresponding to the user, and creates a bucket corresponding to the user.
- Users can also enable service by using the accseskey and secretkey, and encode according to the instruction given in the SDK demo to proceed. Users can submit more requirements such as:
- upload file type, picture video text.
- Average file size range.
- Approximate storage space requirements.
- Whether you need to use CDN. - Whether upload acceleration is required.

Users can view the following information in real-time through the self-service management console

- User space information, including usage, number of files used.
- Historical logs such as bandwidth, traffic, and number of requests used by users.
- Users can view file size distribution, file type distribution, etc.
- Provide billing information, including billing details for used traffic, space, and number of requests.
- BaishanCloud storage service user manual, audio and video conversion, image processing and other documents.
- Key management, password management.

[TOC]

Basic knowledge popularization

The concept of Bucket and Object

- Bucket， we store many files in one bucket.
- Object, we consider each file as an object.
- Users can create buckets, delete buckets, query objects inside buckets, set bucket properties, etc.
- The user can create objects (object is the file uploaded by the user), delete objects, download objects, set the access rights of the objects, etc.
- One example: ````
- Create a bucket, bucket name: baishan_yun, request url: PUT /baishan_yun.
- Upload a file to the bucket of baishan_yun, object name: hello_world, each file is uniquely identified by bucket name and object name, requested url: PUT /baishan_yun/hello_world.
- To download this file, request the url: GET /baishan_yun/hello_world.
- To delete this file, the requested url: DELETE /baishan_yun/hello_world
- To delete the bucket, the requested url: DELETE /baishan_yun ````

Programming interface and user’s guide related

What are the technical advantages of BaishanCloud Storage?

- BaishanCloud Storage is a product that closely combines CDN and cloud storage. BaishanCloud Storage accelerates uploads via CDN edge nodes and downloads via whole CDN networks. Users can also use the cloud storage service separately.
- BaishanCloud Storage is a multi-copy, multi-IDC, distributed architecture that ensures high reliability and availability.
- BaishanCloud Storage provides rich and convenient image processing features that help customers also avoid the cost of duplicate uploads/downloads.
- BaishanCloud Storage is a completely self-developed system, supporting storage needs in various scenarios, and providing higher performance and rich features.
- BaishanCloud Storage adheres to the user-oriented principle and can be customized according to user needs, helping users to use the storage more conveniently and providing 24*7 technical staff online service to help users solve problems quickly.

- BaishanCloud Storage provides a variety of user data import mechanisms to facilitate the migration of data to the storage.
- BaishanCloud Storage follows and is compatible with the Amazon S3 interface; it supports the official Amazon S3 SDK, and users can also use various open-source tools.
- BaishanCloud follows the Amazon S3 REST API interface and strictly adheres to the HTTP protocol to avoid various problems caused by inconsistent interfaces and standards, and the Amazon S3 REST API interface, as the industry standard, has not been modified for more than 10 years and is very stable.
- BaishanCloud supports Amazon S3 ACL access control, which can manage the access permissions of buckets and objects. Each bucket and object have an additional ACL sub-resource, which defines which users or groups will be granted access.

List of languages and SDKs supported by BaishanCloud Storage

- BaishanCloud Storage is compatible with the Amazon S3 interface. It includes all major development languages, such as Python, Php, Browser, Node.js, etc.
- In the document baishancloud_storage_api, there are detailed example descriptions of SDKs for Python, Php, Browser, Node.js and other languages, which can help developers get started quickly. For the rest of the rules, you can refer to the Amazon S3 API documentation.
- Android Documentation:AWS Mobile SDK for Android Documentation SDK Download:aws-android-sdk
- Browser Documentation: Getting Started with the SDK in the Browser SDK Download: Bower
- ISO Documentation: AWS Mobile SDK iOS Developer Guid SDK Download: aws-ios-sdk
- Java Documentation: AWS SDK for Java Developer Guide SDK Download: aws-java-sdk
- .NET Documentation: AWS SDK for .NET Documentation SDK Download: AWSToolsAndSDKForNet.msi
- Node.js Documentation: AWS SDK for JavaScript Documentation SDK Download: Getting Started with the SDK in Node.js
- PHP Documentation: AWS SDK for PHP Documentation SDK Download: Installing via Composer
- Python Documentation: Boto 3 Documentation SDK Download&Description: Boto 3 - The AWS SDK for Python

- Ruby Documentation: AWS SDK for Ruby SDK Download:AWS SDK for Ruby Developer Guide v2.16.8
- Go Documentation: AWS SDK for Go SDK Download:aws-sdk-go
- C++ Documentation: AWS SDK for C++ Documentation SDK Download:aws-sdk-cpp

Is the Cloud Storage Object stored in a directory structure?

A BaishanCloud Storage Object is not a file in the traditional sense, and the storage method is not the same as saving in a file system.

Each Object appears to be a file in a directory, but for cloud storage it is just a directory-like string.

However, this does not affect the way users want to list files from the specified directory:

When sending a request to list files, set the prefix parameter to the name of the directory prefix, you can only list the files in this directory, for example:

- bucket/1.mp4
- bucket/cc/2.mp4
- bucket/cc/3 .mp4
- bucket/dd/ee/ff/3.mp4

When the requested `prefix` is specified as `cc`, it will list:

- 2.mp4
- 3.mp4

This process is as same as listing files from a directory. In addition to simulating directory listing files, it also provides functions such as specifying the maximum number of files to list, specifying to list files after a certain file name (sorted in alphabetical order), and merging files according to common prefixes. For details, please refer to the API documentation.

Things to pay attention to when using:

- Object name should not start with '/'.

  For example: You need to upload a file called file.txt, bucket is `sandbox`.

  - Object is set to `file.txt`, when accessing it should be `http://ss.bscstorage.com/sandbox/file.txt` (recommended)
  - If object is set to `/file.txt`, it should be `http://ss.bscstorage.com/sandbox//file.txt` when accessing (strongly not recommended)
- It is best not to have two consecutive '/' in the object name. For example: You need to upload a file called file.txt in the folder directory, and the bucket is `sandbox`,

- Object is set to `folder/file.txt`, when accessing it should be `http://ss.bscstorage.com/sandbox/folder/file.txt` (recommended method)
- If object is set to `folder//file.txt`, it should be `http://ss.bscstorage.com/sandbox/folder//file.txt` when accessing (strongly not recommended)
- Why does BaishanCloud Storage not recommend two '//' in the URL: some CDN cache servers may combine two consecutive '//' in the URL into one '/' round-trip source, in this case, the Authentication failed or 404

How to delete a directory?

BaishanCloud Storage does not support deleting a directory directly,

Because the Amazon S3 API does not have the concept of directories in the strict sense (there is no directory hierarchy, but it is customary to separate objects by "/" as directories), there is no guarantee for multiple file operations in this case, so it is done by scripts.

This can be done in the following way:

- List all the files in the directory via the List Objects interface.
- Deleting them one by one through the Delete Object interface.
- Deleting a directory.

Please note:

- The list of files returned by the List Objects interface is not all the files in that directory (the list is sorted by filename characters). It may contain the next directory or several directories; so you need to adjust the value of the max-keys parameter, and filter each file to make sure it is a file in that directory. If you set the `prefix` parameter, you can strictly list all files under that `prefix`.
- Deleting all files at once (listing all files and deleting them one by one) does not guarantee that all files in the directory will be cleaned up. `(There may be new uploads during the deletion process.)

Is it possible to use BaishanCloud Storage for whole site backup?

BaishanCloud CDN provides a whole-site backup service, but it is not yet integrated with cloud storage.

How do users import data

BaishanCloud Storage provides multiple ways for users to import data.

- End users use the SDK interface to upload files to the storage.

- End users use the SDK to generate offline download tasks, and the storage's offline download subsystem is responsible for downloading data from the origin and synchronizing it to the storage.
- End users configure the rules for 404 back to the origin, and when end user accesses the bucket, it will automatically go to the origin site to grab the data and synchronize it to the storage.
- End users provide the download list of files, and BaishanCloud Storage will download and synchronize the data to the storage through the file list.
- End users can mail the hard disk.

If the file does not exist in the storage, can I automatically have the storage go to a specific place to pull it?

If the file does not exist in BaishanCloud Storage, you can configure it to go to a specific domain to pull the file.

The file pulling is asynchronous, and the pulling completion time is related to the size of the file being pulled.

The request also returns a 302, and the value of the location response header is the address where the file was pulled.
To use the pull function, you need to provide:

- The origin domain name.
- The mapping rule from request uri to the uri where the file is pulled.
- The validation method for the origin request (optional).
- Whether to enable the HEAD request to the origin.

How to use offline download to synchronize data?

The offline download service supports HTTP protocol, BitTorrent protocol, Magnet URI scheme, etc.

By submitting an offline download task, the offline download service will download the file from the address provided in the task and synchronize it to the storage. Once the synchronization is complete, the result of the operation will be sent to the user.

It supports download task status inquiry, task cancellation, download data corruption check, message transmission, etc. High performance with more than 2000 rps/s.

How to use:

- Use SDK, send request to offline.bscstorage.com, request url: http://offline_domain/, bucket name is the bucket to store the file.
- Send the request for the offline task, e.g.

```
{   "Url":"http://www.abc.com/download/movie.mp4",
    "Key":"movies/2016/movie.mp4",
    "SHA1":"abcdeabcdeabcdeabcdeabcdeabcdeabcdeabcde",
    "MD5":"abcdabcdabcdabcdabcdabcdabcdabcd",
    "CRC32":"abcdef",
    "SuccessCallbackUrl":"http://website/succ/",
    "FailureCallbackUrl":"http://website/fail/",
    }
```

- The user needs to start a service for receiving offline download results.

Does it support multi-file uploads?

Multiple file submissions in a single request are not supported.

If the user wants to submit multiple files in one request for efficiency reasons, it is recommended that the user uses TCP long connections to send multiple requests for uploading files,

.

This avoids establishing multiple connections and is about as efficient as submitting multiple files in a single request.

How to upload large files and what are the limitations?

- There is no limit to the total data capacity or the number of data objects on BaishanCloud Storage. The size of each data object can range from 1 byte to 1 TB.
- The maximum data objects that can be uploaded in a single PUT request is 1TB.
- For large files it is recommended to use the Slice Upload interface for uploading large files in slices, the limits for slice uploads are
    - Maximum 512MB per slice.
    - The maximum number of slices is 2000.
    - The maximum upload file size is 1TB.
- Large file uploads are divided into 3 steps:
    - First initializate interface for slice uploads, the request returns an uploadId, the subsequent 2 steps need this uploadId to identify each slice to upload.
    - The large file is divided into several pieces to be uploaded, and the order of the pieces is identified in the request by the partNmuber parameter.
    - When the upload is complete, the slice completion interface is called to notify storage that the file slice upload is complete.

What are the access control permissions?

- The authorized person can be a user, a group or all users
- For Bucket control permissions:
  - Read only: allows the authorized person to list all objects in the bucket.
  - Write: allows the authorized person to create, overwrite and delete objects in the bucket.
  - Read ACL(Access Control List): Allows the authorized person to read the ACL of the bucket.
  - Write ACL: Allows the authorized person to modify the ACL of the bucket.
  - All permissions: Allows the authorized person to read, write, read ACL, and write ACL of the storage bucket.
- For Object control permissions:
  - Read: Allows the authorized person to read the data and meta information of the object.
  - Read ACL(Access Control List): allows the authorized person to read the ACL of the object.
  - Write ACL: allows the authorized person to modify the ACL of the object.
  - All permissions: allow the authorized person to read, read ACL, write ACL on the object.

Is it possible to modify bucket and object without signature (anonymous access)?

Normal operations require a signature to verify user permissions, but it is possible to set a common attribute that allows users to operate without a signature.

- By setting the bucket permissions to public-read, any user can list the files under the bucket and request them without a signature.
- After setting the bucket permission to public-read-write, any user can list the files under this bucket and upload files to this bucket, and the request does not require a signature.
- When uploading a file, after specifying object permissions as public-read, any user can download the file without a signature, which is called anonymous download.
- When a user accesses the storage through a CDN request, the request does not require a signature to access the files on the storage because the CDN nodes are configured with full permissions.

Does storage support decompression of user's file packages?

- It is not supported for now, however, it can be developed if required.

Does it support restore downloads?

- Support, use HTTP1.1 protocol (RFC2616) Range and Content-Range header field to restore download, specific header field settings refer to rfc2616

Video processing

Is there a requirement for origin file format for transcoding?

There are no requirements, basically all formats are supported.

Does the transcoding template describe the input file or the output file?

The transcoding template is to configure the format and other information of the output file, there is no need to configure anything for the input file.

What does transcoding templates include?

- Encapsulation formats, flv, mp4, ts, etc.
- Video encoding format, h265, h264, mpeg2, etc.
- Video bitrate
- Video resolution
- Video keyframe interval
- Video frame rate
- Audio encoding format, aac, mp2, mp3, etc.
- Audio bitrate
- Number of audio channels

What output formats are supported:

- flv
- fmp4
- mp4
- ts
- mp3
- wma
- wmv

- mp2
- aac
- gif

Does it support video slicing?

Yes, you can create a transcoding task and specify the duration of the slice segment, provided that the package format ts of the template configuration is used.

Does it support video watermark?

Yes, the watermark location information is configured in the transcoding template, and you can create a transcoding task and add a watermark image.

Are video clips supported?

Yes, you can create a transcoding task and specify the start time and duration.

Does hls slice support AES128 encryption?

Support, create transcoding and provide encrypted key, decrypted url, and encrypted iv vector (optional).

Does it support 4K transcoding?

Support, you can configure 4k resolution in the template

automatic transcoding

Automatic transcoding means that a transcoding task is automatically created for this file system while uploading a file.

The path where the output file is stored, for example, the source file is a/b/xx.mp4

If the configuration is consistent with the source file path, the file suffix is configured as foo.ts, and the key in the storage is a/b/xxfoo.ts

If there is no configuration and the source file path is consistent, the file suffix is configured as foo.ts, and the key in the storage is xxfoo.ts

How long does it take to transcode a 1h video?

Transcoding takes about 45 – 60 minutes. If the video encoding and audio encoding remain unchanged (Also known as transcoding, which only the suffix name has changed), it will take 3 - 5 minutes. This is only the estimated time, high resolution and special format time will take longer.

Can the m3u8 file generated by video slicing be played directly through cloud storage?

Normally it cannot be played. The m3u8 file stores a list of a bunch of small ts files. When playing, the player needs to download the small ts files from the storage, but the download needs to be signed.

Set all small ts files to be anonymous and accessible, and they can be played.

Through CDN, when cloud storage is set as the origin, the file can be played.

Manually created a transcoding task, but can't find the target file after completion?

It is possible that the output file prefix is not specified, and the target file is placed in the bucket root directory

Image processing

Does BaishanCloud Storage support real-time image conversion, rotation, scaling, cutting, and watermarking?

- Yes, we support those functions, please see imgx_manual documentation for details.
- The processed image is generated and cached, and the next request will not be repeated, and the default cache period is 7 days.
- If a CDN acceleration is configured, the processed images will be automatically pushed to the CDN node.
- If the original image is modified, you can use the v command to regenerate the image and link.

Is it possible to let the storage automatically determine that the user-agent triggers a specific image cropping output?

Not supported currently, please contact your account team to submit development request if needed.

Is there a video transcoding function?

- Support conversion between flv, mp4, mov and other formats
- Support for reducing bit rate, resolution, modifying gop (key frame interval) and other functions
- Support video screenshots.

This feature is currently under development and will be provided in the future. Please contact the support team for more details.

User management storage resource related, management console

What information can storage users view through the background?

The information that can be viewed in the background includes:

- User space information, including usage, number of files used.
- You can view the user's bandwidth, traffic, number of requests, etc.
- You can view file size distribution, file type distribution, etc.
- Provide billing information, including billing details for used traffic, space, and number of requests.
- Provide documents such as image processing to help users use.
- Provides key management.

Whether the storage console has the function of searching for files. For example, the full path of a file in a bucket is abc/def/123.png. Can you use the file name 123.png to search for the file without knowing the full path? ?

Currently, there is no search function developed in the console. You can use the head command to find files under each bucket.

Can I count the number of files under a bucket?

The number of files can be viewed from the number of files in the bucket in the database.

Does Baishan Cloud Storage provide log download?

It is currently not supported from the console, please contact the support team for assistance.

Performance, Reliability, Availability

Does Baishan Cloud Storage do upload acceleration?

Yes, Baishan cloud storage uses its robust network with the edge nodes of CDN to improve the upload speed.

When the user requests accelerated upload of the domain name, the DNS server intelligently returns the IP of the CDN edge node closest to the user to the user.

The user uploads the file to the nearest edge node, and the edge node transmits the data to the storage center through the optimal path.

Compared with the accelerated upload by the edge node and the direct upload to the storage center, the average speed is more than doubled, which greatly improves the upload experience.

Using upload acceleration is very simple, you only need to use acceleration to upload the domain name.

Does Baishan Cloud Storage integrate CDN support?

Baishan cloud storage and Baishan CDN are perfectly integrated, application for storage and CDN can be carried out at the same time, and there is a complete configuration process

- CDN upload acceleration
- CDN supports virtual domain name back-to-origin
- CDN supports signature back-to-source to protect user data security
- At the same time, Baishan Cloud Storage also supports third-party CDN back-to-origin
- How to configure CDN and cloud storage:

```
If you want to test the download acceleration through CDN, you need to change

For example, www.qq.com.i.qingcdn.com, the CDN returns to ss.bscstorage.com.


Suggestions for creating buckets:

1. Create a bucket with the same name as the user domain name.

For example, if the user website domain name is www.qq.com, then create a buck

2. User-defined bucket name:

For example, if the user website domain name is www.qq.com, and the user wants
```

Is HTTPS supported? How to use?

- Support HTTPS method.
- If it is a user domain name, first, the user domain name needs to have ICP license, and the user needs to apply for an SSL certificate for the domain name, and then upload the certificate to BaishanCloud console, please contact support team if you need any assistance with aquiring SSL certificate.
- If you use the domain name provided by Baishan Cloud Storage, for example: https://ss.bscstorage.com, Note: The bucket must be placed in the url request, not the domain name.

Is encrypted storage supported?

Not currently supported

How reliable is it?

- Baishan Cloud Storage ensures reliability up to 99.999% through mechanisms such as multiple data center, multiple cached files, and fast failover.

Pre-sales support, POC testing, etc.

Is there an environment for quick POC testing? What is the application process?

Currently it is manually reviewed and opened. Please provide the following information when applying:

- The full company name of the applicant client.
- The user is required to provide an email address (login email).
- The initial password for the user to log in to the storage console.
- In order to better cooperate with the test, the user is also required to provide the following information:
  - Upload file type, image video text.
  - Average file size range.
  - Approximate storage space requirements.
  - Whether you need to use CDN.
  - Whether upload acceleration is required.
- BaishanCloud support team will create a user, generate the accseskey and secretkey corresponding to the user, and create a bucket corresponding to the user.

- Users can use accseskey and secretkey, and encode according to the demo given in the SDK.

How to apply for upload acceleration?

Use the upload acceleration domain name and call the SDK for uploading files.

How to apply for storage and CDN service at the same time?

- CDN user application address: http://www.qingcdn.com/account/index/login
- Cloud storage users, manual application.

How does the origin set the cache time?

BaishanCloud Storage can set the settings of HTTP request headers such as Cache-Control of stored files and send the cache settings together with HTTP access requests.

BaishanCloud CDN can execute the cache policy according to the Cache-Control setting in the HTTP request header returned by the origin site.

Related to third-party service integration

Is there an integrated text editor for uploading images?

Some customers have proposed the whole page upload function, but this function is not supported currently.

Daily use related

Does BaishanCloud Storage provide a web management interface for users to manage their own files?

Provide, management interface login address: https://cwn-ss.portal.baishancloud.com

How to upload files? Is it possible to upload via FTP or rsync? Is there a client can be used?

BaishanCloud Storage follows and is compatible with Amazon S3 REST API.

There are many mature and reliable SDKs and command-line tools in the community, which are compatible with Amazon S3 REST API, so they can be used directly, which can completely save the development work of the interface layer.

For example the popular:

- s3cmd
- boto

Baishan cloud storage does not support FTP or rsync protocol. Because amazon s3 has more fine-grained control over file permissions.

Graphical FTP alternatives can use cyberduck and crossftp for Mac or s3browser and crossftp for Windows.

Alternatives to command line ftp or rsync can use s3cmd

or aws cli

To support mounting a bucket on a Linux host, so that files in the bucket can be accessed like local files, the following third-party tools can be used

- s3fs
- riofs
- googys

SDK Related

Error when uploading file using java SDK： Unable to verify integrity of data upload. Client calculated content hash (contentMD5: E+6KS0B2pNPJ272XbG92fw== in base 64) didn't match hash (etag: 359740e5918c395b6e35a2c612582e42 in hex) calculated by Amazon S3

When uploading files, the Java SDK uses the "ChunkedEncoding" method by default. We do not support this method at the moment. You can disable this upload method through parameter settings. Please refer to the following example.

```
BasicAWSCredentials awsCreds = new BasicAWSCredentials(
        accessKey, secretKey);
AmazonS3 s3 = new AmazonS3Client(awsCreds);

s3.setS3ClientOptions(S3ClientOptions.builder().disableChunkedEncoding().build
```

Error when uploading file using android SDK：Unable to verify integrity of data upload. Client calculated content hash didn't match hash calculated by Amazon S3. You may need to delete the data stored in Amazon S3

When uploading files, the android SDK uses the "ChunkedEncoding" method by default. We do not support this method for the time being. Although the android SDK provides an interface to disable this method, it still cannot take effect after setting it to disable, so it needs to be signed by using v2. to circumvent this problem. "ChunkedEncoding" is only used for v4 signatures.

To set the method to use v2 signature, please refer to the following code.

```
AWSCredentials credentials = new BasicAWSCredentials(
        accessKey, secretKey);

ClientConfiguration configuration = new ClientConfiguration();
configuration.setSignerOverride("S3SignerType");

AmazonS3 s3 = new AmazonS3Client(credentials, configuration);
```

When uploading and downloading files using the ios SDK, an error occurs: Code: 403; Error Code: SignatureDoesNotMatch; Request ID: 23e338d9-1706-1410-3759-a0369fd80cca

The iOS SDK will determine the service type according to the domain name specified by the endpoint. For example, if the domain name starts with 's3' (s3.amazonaws.com), it means that the request is for s3 service, and the corresponding signature method is used. Therefore, when specifying a non-AWS standard When the domain name is used, it will result in the use of an incorrect signature method. In addition, there are few configurable parameters in the ios SDK, and the behavior of the SDK can only be changed by modifying the source code. Source code github address: 'https://github.com/aws/aws-sdk-ios.git'. A new line needs to be inserted at line 319 of the file AWSCore/Authentication/AWSSignature.m: '[request setValue:contentSha256 forHTTPHeaderField:@"x-amz-content-sha256"];'. Then execute the following command in the source code root directory to compile the source code:

```
chmod +x Scripts/SdkPackage.sh
sh Scripts/Package.sh
```

Since you only need to recompile AWSCore, you can edit the Scripts/Package.sh file and delete the code for compiling other modules. After deletion, the end of the file is as follows:

```
if [ -x "Scripts/SdkPackage.sh" ]; then
    Scripts/SdkPackage.sh AWSCore
fi
```

The compiled framework is located in the builtFramework/framework directory. Use the newly compiled AWSCore.framework to replace the original one in the project.

Downloading AWS SDK from Maven Repository is too slow and easily lead to disconnection. You can add the following configuration to pom.xml to replace the mirror.

```
<repositories>
    <repository>
        <id>my-repo1</id>
        <name>your custom repo</name>
        <url>http://mirrors.redv.com/maven2</url>
    </repository>
</repositories>
```

How to set up signature version for SDK

java SDK

java SDK default using signature version 4, to use signature version 2, please add the following script:

```
import com.amazonaws.ClientConfiguration;

ClientConfiguration clientconfiguration = new ClientConfiguration();
clientconfiguration.setSignerOverride("S3SignerType");
AmazonS3 client = new AmazonS3Client(awsCreds, clientconfiguration);
```

python SDK

python SDK(boto3) default using signature version 2, to use signature version 4, please add the following script:

```
from botocore.client import Config

config = Config(signature_version='s3v4')
client = boto3.client(
    's3',
    aws_access_key_id = access_key,
    aws_secret_access_key = secret_key,
    config = config,
    region_name = 'us-east-1',
    endpoint_url = 'http://bscstorage.com',
)
```

php SDK

php SDK only support signature version 4, signature version 2 not supported.

go SDK

go SDK only support signature version 4, signature version 2 not supported.

JavaScript SDK

JavaScript SDK default using signature version 2, to use signature version 4, please add the following script:

```
AWS.config.signatureVersion = 'v4';
```

Obeject Storage

Instance（Compatible Amazon SDK）

Python Demo

Install the AWS Python client – boto3

```
pip install boto3
```

Initialization & account and domain setup

```
import boto3
from boto3.s3.transfer import TransferConfig

cli = boto3.client(
    's3',
    aws_access_key_id='ziw5dp1alvty9n47qksu', #please replace with your access
    aws_secret_access_key='V+ZTZ5u5wNvXb+KP5g0dMNzhMeWe372/yRKx4hZV', #please
    endpoint_url='http://ss.bscstorage.com'
)
```

File operation API

File Upload

Using the put_object interface

ACL can set as：'private' or 'public-read' or 'public-read-write' or 'authenticated-read'

```
resp = cli.put_object(
    ACL='public-read',
    Bucket='test-bucket-xxx',
    Key='test-key-xxx',
    ContentType='image/jpeg', # please replace with applicable content type
    Body='the content of the file as a string'
)
```

Using upload_file interface (It suits to upload the large File, it supports dividing the File into different blocks automatically and uploading those blocks simultaneously.)

```
config = TransferConfig(
    multipart_threshold=30 * 1024 * 1024,
    multipart_chunksize=8 * 1024 * 1024,
    max_concurrency=10
)
resp = cli.upload_file(
    '/root/test.mp4',
    'test-bucket-xxx',
    'test-key-xxx',
    ExtraArgs={
        'ContentType': 'image/jpeg', # please replace with applicable content
        'ACL': 'private',
    },
    Config=config
)
```

File Copy

Copy all files in the source bucket prefixed with `aa` to the target bucket

```
marker = ''

while True:
    resp = s3.list_objects(
        Bucket='src-bucket',
        Prefix='aa',
        Marker=marker,
    )

    if 'Contents' not in resp:
        break

    for content in resp['Contents']:
        s3.copy_object(Bucket='dst-bucket', Key=content['key'], CopySource='/%

    marker = resp['Contents'][-1]['Key']
```

File Download

```
resp = cli.get_object(
    Bucket='test-bucket-xxx',
    Key='test-key-xxx'
)
```

Get File URL

获取已签名的URL用来下载文件，可通过参数ExpiresIn设置签名过期时间。

```
url = cli.generate_presigned_url(
    'get_object',
    Params={
        'Bucket': 'test-bucket-xxx',
        'Key': 'test-key-xxx'
    },
    ExpiresIn=60
)
print url
```

File Delete

```
resp = cli.delete_object(
    Bucket='test-bucket-xxx',
    Key='test-key-xxx'
)
```

Get File ACL

```
resp = cli.get_object_acl(
    Bucket='test-bucket-xxx',
    Key='test-key-xxx'
)
```

Set File ACL

Using the pre-defined ACL

Support pre-defined ACL：'private', 'public-read', 'public-read-write' 或 'authenticated-read'

```
resp = cli.put_object_acl(
    ACL='public-read',
    Bucket='test-bucket-xxx',
    Key='test-key-xxx'
)
```

Using the custom ACL

Permission includes：'FULL_CONTROL', 'WRITE', 'WRITE_ACP', 'READ', 'READ_ACP'

```
resp = cli.put_object_acl(
    AccessControlPolicy={
        'Grants': [
            {
                'Grantee': {
                    'ID': 'user_foo', # 请替换为真实存在的用户
                    'Type': 'CanonicalUser',
                },
                'Permission': 'WRITE',
            },
            {
                'Grantee': {
                    'ID': 'your-user-name',
                    'Type': 'CanonicalUser',
                },
                'Permission': 'FULL_CONTROL',
            },
        ],
        'Owner': {
            'ID': 'your-user-name',
        },
    },
    Bucket='test-bucket-xxx',
    Key='test-key-xxx'
)
```

Bucket operation API

Bucket Create

ACL can be set as： 'private' or 'public-read' or 'public-read-write' or 'authenticated-read'

```
resp = cli.create_bucket(
    ACL='public-read',
    Bucket='test-bucket-xxx'
)
```

List Bucket File (List all the files contained on the bucket. The max number of returning files each time is 1000)

```
resp = cli.list_objects(
    Bucket='test-bucket-xxx',
    Prefix='',
    Marker='',
)
```

List all the files contained in the bucket

```
marker = ''

while True:
    resp = s3.list_objects(
        Bucket='test-bucket-xxx',
        Marker=marker,
    )

    if 'Contents' not in resp:
        break

    for content in resp['Contents']:
        print 'key: %s, size: %d' % (content['Key'], content['Size'])

    marker = resp['Contents'][-1]['Key']
```

Bucket Delete

```
resp = cli.delete_bucket(
    Bucket='test-bucket-xxx'
)
```

Get Bucket ACL

```
resp = cli.get_bucket_acl(
    Bucket='test-bucket-xxx'
)
```

Set Bucket ACL

Using the pre-defined ACL

Support pre-defined ACL： 'private', 'public-read', 'public-read-write' or 'authenticated-read'

```
resp = cli.put_bucket_acl(
    ACL='public-read',
    Bucket='test-bucket-xxx',
)
```

Using the custom ACL

Permission includes： 'FULL_CONTROL', 'WRITE', 'WRITE_ACP', 'READ', 'READ_ACP'

```
resp = cli.put_bucket_acl(
    AccessControlPolicy={
        'Grants': [
            {
                'Grantee': {
                    'ID': 'user_foo', # 请替换为真实存在的用户
                    'Type': 'CanonicalUser',
                },
                'Permission': 'WRITE',
            },
            {
                'Grantee': {
                    'ID': 'your-user-name',
                    'Type': 'CanonicalUser',
                },
                'Permission': 'FULL_CONTROL',
            },
        ],
        'Owner': {
            'ID': 'your-user-name',
        },
    },
    Bucket='test-bucket-xxx'
)
```

Service operation API

List all buckets

```
resp = cli.list_buckets()
```

AWS Official SDK aws-sdk-python

API doc api-reference

PHP Demo

System Requirement:

- PHP>=5.5.0

Install the AWS SDK for PHP

```
curl -O http://docs.aws.amazon.com/aws-sdk-php/v3/download/aws.phar
```

Initialization & Set the account information and the domain name

Request Syntax

```
 require 'aws.phar';
 $cli = new Aws\S3\S3Client([
     'version' => 'latest',
    'region' => 'us-east-1',
     'credentials' => [
         'key' => 'z2qutjf718d0i9gw6skc',  //Please fill in it with your own a
        'secret' => 'SEQgcc1ppH7uXPG4ZPIcrCv2cWz8grcReMfFABCn', // Please fill
    ],
    'endpoint' => 'http://ss.bscstorage.com',
]);
```

File Operation API

File Upload

The permitted values of ACL are 'private', 'public-read', 'public-read-write', and 'authenticated-read'. If the file already exists in the storage, the developer could use 'Body' to call it. If the file is on disk, the developer could use 'SourceFile' to name the file. 'Body' and 'SourceFile' cannot be used simultaneously.

```
$resp = $cli->putObject([
    'ACL' => 'public-read',
    'Bucket' => 'test-bucket-xxx',
    'Key' => 'test-key-xxx',
    'ContentType' => 'image/jpeg',  //Please fill in it with the proper file t
    'Body' => 'file content as a string',
    //'SourceFile' => '/root/test.jpg',
]);
```

File Download

```
$resp = $cli->getObject([
    'Bucket' => 'test-bucket-xxx',
    'Key' => 'test-key-xxx',
]);
```

Get File URL

Get the pre-signed URL to download the File, and the developer could set an expired time.

```
$cmd = $cli->getCommand('GetObject', [
    'Bucket' => 'test-bucket-xxx',
    'Key' => 'test-key-xxx',
]);
$req = $cli->createPresignedRequest($cmd, '+10 seconds');
$url = (string) $req->getUri();
```

File Delete

```
$resp = $cli->deleteObject([
    'Bucket' => 'test-bucket-xxx',
    'Key' => 'test-key-xxx',
]);
```

Get File ACL

```
$resp = $cli->getObjectAcl([
    'Bucket' => 'test-bucket-xxx',
    'Key' => 'test-key-xxx',
]);
```

Set File ACL

Using the pre-defined ACL

The permitted values of ACL are private, public-read, public-read-write, and authentication-read.

```
$resp = $cli->putObjectAcl([
    'ACL' => 'public-read-write',
    'Bucket' => 'test-bucket-xxx',
    'Key' => 'test-key-xxx',
]);
```

Using the custom ACL

The permitted values of Permission are 'FULL_CONTROL', 'WRITE', 'WRITE_ACP', 'READ', and 'READ_ACP'

```
$resp = $cli->putObjectAcl([
    'AccessControlPolicy' => [
        'Grants' => [
            [
                'Grantee' => [
                    'ID' => 'user_foo', //Please fill it with the existing use
                    'Type' => 'CanonicalUser',
                ],
                'Permission' => 'WRITE',
            ],
            [
                'Grantee' => [
                    'ID' => 'your-user-name',
                    'Type' => 'CanonicalUser',
                ],
                'Permission' => 'FULL_CONTROL',
            ],

        ],
        'Owner' => [
            'ID' => 'your-user-name',
        ],
    ],
    'Bucket' => 'test-bucket-xxx',
    'Key' => 'test-key-xxx',
]);
```

Bucket operation API

Bucket Create

The permitted values of ACL are 'private', 'public-read', 'public-read-write', and 'authentication-read'.

```
$resp = $cli->createBucket([
    'Bucket' => 'test-bucket-xxx',
    'ACL' => 'public-read',
]);
```

List all the files contained on the bucket. The max number of returning files at the same time is 1000

```
$resp = $cli->listObjects([
    'Bucket' => 'test-bucket-xxx',
    'Prefix' => '',
    'Marker' => '',
]);
```

List all the files contained on the bucket.

```
$marker = '';
while (true):
    $resp = $cli->listObjects([
    'Bucket' => 'test-bucket-xxx',
    'Marker' => $marker,
    ]);

    if($resp['Contents'] == NULL)
    {
        break;
    }
    foreach($resp['Contents'] as $content)
    {
        var_dump($content['Key']);
        $marker = $content['Key'];
    }

endwhile;
```

Bucket Delete

```
$resp = $cli->deleteBucket([
    'Bucket' => 'test-bucket-xxx',
]);
```

Get Bucket ACL

```
$resp = $cli->getBucketAcl([
    'Bucket' => 'test-bucket-xxx',
]);
```

Set Bucket ACL

Using the pre-defined ACL

The permitted values of ACL are 'private', 'public-read', 'public-read-write', and 'authentication-read'.

```
$resp = $cli->putBucketAcl([
    'ACL' => 'public-read-write',
    'Bucket' => 'test-bucket-xxx',
]);
```

Using the custom ACL

The permitted values of Permission are 'FULL_CONTROL', 'WRITE', 'WRITE_ACP', 'READ', and 'READ_ACP'

```
$resp = $cli->putBucketAcl([
    'AccessControlPolicy' => [
    'Grants' => [
            [
                'Grantee' => [
                    'ID' => 'user\_foo', // Please fill it with the existing u
                    'Type' => 'CanonicalUser',
                ],
                'Permission' => 'WRITE',
            ],
            [
                'Grantee' => [
                    'ID' => 'your-user-name',
                    'Type' => 'CanonicalUser',
                ],
                'Permission' => 'FULL\_CONTROL',
            ],

        ],
        'Owner' => [
            'ID' => 'your-user-name',
        ],
    ],
    'Bucket' => 'test-bucket-xxx',
]);
```

Service Operation API

Buckets List (List all the Buckets of your service)

```
$resp = $cli->listBuckets([
]);
```

Note: 1.Official AWS SDK for PHP: aws-sdk-php 1.API documentation: api-reference

Python Demo

Install the AWS Python client – boto3

```
pip install boto3
```

Initialization & account and domain setup

```
import boto3
from boto3.s3.transfer import TransferConfig

cli = boto3.client(
    's3',
    aws_access_key_id='ziw5dp1alvty9n47qksu', #please replace with your access
    aws_secret_access_key='V+ZTZ5u5wNvXb+KP5g0dMNzhMeWe372/yRKx4hZV', #please
    endpoint_url='http://ss.bscstorage.com'
)
```

File operation API

File Upload

Using the put_object interface

ACL can set as：'private' or 'public-read' or 'public-read-write' or 'authenticated-read'

```
resp = cli.put_object(
    ACL='public-read',
    Bucket='test-bucket-xxx',
    Key='test-key-xxx',
    ContentType='image/jpeg', # please replace with applicable content type
    Body='the content of the file as a string'
)
```

Using upload_file interface (It suits to upload the large File, it supports dividing the File into different blocks automatically and uploading those blocks simultaneously.)

```
config = TransferConfig(
    multipart_threshold=30 * 1024 * 1024,
    multipart_chunksize=8 * 1024 * 1024,
    max_concurrency=10
)
resp = cli.upload_file(
    '/root/test.mp4',
    'test-bucket-xxx',
    'test-key-xxx',
    ExtraArgs={
        'ContentType': 'image/jpeg', # please replace with applicable content
        'ACL': 'private',
    },
    Config=config
)
```

File Copy

Copy all files in the source bucket prefixed with `aa` to the target bucket

```
marker = ''

while True:
    resp = s3.list_objects(
        Bucket='src-bucket',
        Prefix='aa',
        Marker=marker,
    )

    if 'Contents' not in resp:
        break

    for content in resp['Contents']:
        s3.copy_object(Bucket='dst-bucket', Key=content['key'], CopySource='/%

    marker = resp['Contents'][-1]['Key']
```

File Download

```
resp = cli.get_object(
    Bucket='test-bucket-xxx',
    Key='test-key-xxx'
)
```

Get File URL

获取已签名的URL用来下载文件，可通过参数ExpiresIn设置签名过期时间。

```
url = cli.generate_presigned_url(
    'get_object',
    Params={
        'Bucket': 'test-bucket-xxx',
        'Key': 'test-key-xxx'
    },
    ExpiresIn=60
)
print url
```

File Delete

```
resp = cli.delete_object(
    Bucket='test-bucket-xxx',
    Key='test-key-xxx'
)
```

Get File ACL

```
resp = cli.get_object_acl(
    Bucket='test-bucket-xxx',
    Key='test-key-xxx'
)
```

Set File ACL

Using the pre-defined ACL

Support pre-defined ACL：'private', 'public-read', 'public-read-write' 或 'authenticated-read'

```
resp = cli.put_object_acl(
    ACL='public-read',
    Bucket='test-bucket-xxx',
    Key='test-key-xxx'
)
```

Using the custom ACL

Permission includes：'FULL_CONTROL', 'WRITE', 'WRITE_ACP', 'READ', 'READ_ACP'

```
resp = cli.put_object_acl(
    AccessControlPolicy={
        'Grants': [
            {
                'Grantee': {
                    'ID': 'user_foo', # 请替换为真实存在的用户
                    'Type': 'CanonicalUser',
                },
                'Permission': 'WRITE',
            },
            {
                'Grantee': {
                    'ID': 'your-user-name',
                    'Type': 'CanonicalUser',
                },
                'Permission': 'FULL_CONTROL',
            },
        ],
        'Owner': {
            'ID': 'your-user-name',
        },
    },
    Bucket='test-bucket-xxx',
    Key='test-key-xxx'
)
```

Bucket operation API

Bucket Create

ACL can be set as： 'private' or 'public-read' or 'public-read-write' or 'authenticated-read'

```
resp = cli.create_bucket(
    ACL='public-read',
    Bucket='test-bucket-xxx'
)
```

List Bucket File (List all the files contained on the bucket. The max number of returning files each time is 1000)

```
resp = cli.list_objects(
    Bucket='test-bucket-xxx',
    Prefix='',
    Marker='',
)
```

List all the files contained in the bucket

```
marker = ''

while True:
    resp = s3.list_objects(
        Bucket='test-bucket-xxx',
        Marker=marker,
    )

    if 'Contents' not in resp:
        break

    for content in resp['Contents']:
        print 'key: %s, size: %d' % (content['Key'], content['Size'])

    marker = resp['Contents'][-1]['Key']
```

Bucket Delete

```
resp = cli.delete_bucket(
    Bucket='test-bucket-xxx'
)
```

Get Bucket ACL

```
resp = cli.get_bucket_acl(
    Bucket='test-bucket-xxx'
)
```

Set Bucket ACL

Using the pre-defined ACL

Support pre-defined ACL： 'private', 'public-read', 'public-read-write' or 'authenticated-read'

```
resp = cli.put_bucket_acl(
    ACL='public-read',
    Bucket='test-bucket-xxx',
)
```

Using the custom ACL

Permission includes： 'FULL_CONTROL', 'WRITE', 'WRITE_ACP', 'READ', 'READ_ACP'

```
resp = cli.put_bucket_acl(
    AccessControlPolicy={
        'Grants': [
            {
                'Grantee': {
                    'ID': 'user_foo', # 请替换为真实存在的用户
                    'Type': 'CanonicalUser',
                },
                'Permission': 'WRITE',
            },
            {
                'Grantee': {
                    'ID': 'your-user-name',
                    'Type': 'CanonicalUser',
                },
                'Permission': 'FULL_CONTROL',
            },
        ],
        'Owner': {
            'ID': 'your-user-name',
        },
    },
    Bucket='test-bucket-xxx'
)
```

Service operation API

List all buckets

```
resp = cli.list_buckets()
```

AWS Official SDK aws-sdk-python

API doc api-reference

JavaScript Demo

```
<html>
<!--Upload the data in the input text box as a file, list the files that have
<!--It is not recommended to put the account information into the browser-side
<textarea id="data"></textarea>
<button id="upload-button">Upload to S3</button>
<div id="results"></div>
<script src='https://sdk.amazonaws.com/js/aws-sdk-2.5.0.min.js'></script>
<script type="text/javascript">
    AWS.config.update({
        accessKeyId: 'ziw5dp1alvty9n47qksu',    <!--Please replce with your ow
        secretAccessKey: 'V+ZTZ5u5wNvXb+KP5g0dMNzhMeWe372/yRKx4hZV'    <!--Ple
    });
    AWS.config.region = 'us-west-1';
    AWS.config.endpoint = 'http://ss.bscstorage.com';
    AWS.config.s3ForcePathStyle = true

    var bucket_name = 'test-bucket'    <!--Please replace with your own bucket

    var s3 = new AWS.S3({
        params: {
            Bucket: bucket_name
        }
    });

    var textarea = document.getElementById('data');
    var button = document.getElementById('upload-button');
    var results = document.getElementById('results');

    button.addEventListener('click', function() {
        results.innerHTML = '';

        var params = {
            Key: 'data.txt',
            Body: textarea.value
        };
        s3.upload(params, function(err, data) {
            results.innerHTML = err ? 'ERROR!' : 'SAVED.';

            s3.listObjects({
                Bucket: bucket_name
            }, function(err, data) {
                if (err) {
                    console.log(err);
                } else {
                    console.log(data);
                }

            });
            s3.getObject({
                Bucket: bucket_name,
                Key: 'data.txt'
            }, function(err, data) {
                if (err) {
                    console.log(err);
                } else {
                    console.log(data);
                }
            });

        });

    }, false);
</script>
</html>
```

```
<html>
<!--Example of uploading a user's local file and generating the URL of the fil
<!--It is not recommended to put the account information into the browser-side
<input type="file" id="file-chooser" />
<button id="upload-button">Upload to S3</button>
<div id="results"></div>
<div id="signed_url"></div>

<script src='https://sdk.amazonaws.com/js/aws-sdk-2.5.0.min.js'></script>
<script type="text/javascript">

  <!--Please replce with your own access key and secret key-->
  AWS.config.update({accessKeyId: 'ziw5dp1alvty9n47qksu', secretAccessKey: 'V+
  AWS.config.region = 'us-west-1';
  AWS.config.endpoint = 'http://ss.bscstorage.com';
  AWS.config.s3ForcePathStyle = true

  var bucket_name = 'test-bucket'    <!--Please replace with your own bucket n

  var s3 = new AWS.S3({params: {Bucket: bucket_name}});

  var fileChooser = document.getElementById('file-chooser');
  var button = document.getElementById('upload-button');
  var results = document.getElementById('results');
  var signed_url = document.getElementById('signed_url')

  button.addEventListener('click', function() {
    var file = fileChooser.files[0];
    if (file) {
      results.innerHTML = '';
      signed_url.innerHTML = '';

      var params = {Key: file.name, ContentType: file.type, Body: file};
      s3.upload(params, function (err, data) {
        var params_get = { Bucket: bucket_name, Key: file.name, Expires: 60 };
        var url = s3.getSignedUrl('getObject', params_get, function(err, url)
            signed_url.innerHTML = err ? 'ERROR!' : 'retrive the file use this
            });
        results.innerHTML = err ? 'ERROR!' : 'UPLOADED.';
      });
    } else {
      results.innerHTML = 'Nothing to upload.';
    }
  }, false);
</script>
</html>
```

AWS Official SDK aws-sdk-browser

Node.js Demo

Install the AWS SDK for JavaScript in Node.js

```
npm install aws-sdk
```

Initialization & Set the account information and the domain name

```
var AWS = require('aws-sdk');
var fs = require('fs');

AWS.config.accessKeyId = 'ziw5dp1alvty9n47qksu'; // Please change the example
AWS.config.secretAccessKey = 'V+ZTZ5u5wNvXb+KP5g0dMNzhMeWe372/yRKx4hZV'; // Pl
AWS.config.region = 'us-west-1';
AWS.config.endpoint = 'http://ss.bscstorage.com';
AWS.config.s3ForcePathStyle = true

var s3 = new AWS.S3();
```

File Operation API

File Upload

Using ‘putObject’ API to upload the File

The permitted values of ACL are private, public-read, public-read-write, and authenticated-read.

```
var params = {
    Bucket: 'test-bucket-xxx',
    Key: 'test-key-xxx',
    ACL: 'public-read',
    Body: new Buffer('blablabla')
};
s3.putObject(params, function(err, data) {
    if (err) console.log(err, err.stack);
    else console.log(data);
});
```

Using the 'upload' interface (It suits to upload the large File, it supports dividing the File into different blocks automatically and uploading those blocks simultaneously.)

```
var file_stream = fs.createReadStream('/root/test.mp4')
var params = {
    Bucket: 'test-bucket-xxx',
    Key: 'test-key-xxx',
    Body: file_stream,
    ACL: 'public-read'
}
var options = {partSize: 10 * 1024 * 1024, queueSize: 10}
s3.upload(params, options, function(err, data) {
    if (err) console.log(err, err.stack);
    else console.log(data);
});
```

File Download

```
var params = {
    Bucket: 'test-bucket-xxx',
    Key: 'test-key-xxx'
};
s3.getObject(params, function(err, data) {
    if (err) console.log(err, err.stack);
    else console.log(data);
});
```

Get File URL

Get the pre-signed URL to download the File, and the developer could set an expired time.

```
var params = {
    Bucket: 'test-bucket-xxx',
    Key: 'test-key-xxx',
    Expires: 60
};
var url = s3.getSignedUrl('getObject', params);
console.log('The URL is', url);
```

File Delete

```
var params = {
    Bucket: 'test-bucket-xxx',
    Key: 'test-key-xxx'
};
s3.deleteObject(params, function(err, data) {
    if (err) console.log(err, err.stack);
    else console.log(data);
});
```

Get File ACL

```
var params = {
    Bucket: 'test-bucket-xxx',
    Key: 'test-key-xxx'
};
s3.getObjectAcl(params, function(err, data) {
    if (err) console.log(err, err.stack);
    else console.log(data);
});
```

Set File ACL

Using pre-defined ACL

The permitted values of ACL are private, public-read, public-read-write, and authentication-read

```
var params = {
    Bucket: 'test-bucket-xxx',
    Key: 'test-key-xxx',
    ACL: 'public-read'
};
s3.putObjectAcl(params, function(err, data) {
    if (err) console.log(err, err.stack);
    else console.log(data);
});
```

Using custom ACL

The permitted values of Permission are FULL_CONTROL, WRITE, WRITE_ACP, READ, and READ_ACP

```
var params = {
    Bucket: 'test-bucket-xxx',
    Key: 'test-key-xxx',
    AccessControlPolicy: {
        Grants: [
            {
                Grantee: {
                    Type: 'CanonicalUser',
                    ID: 'user-foo'
                },
                Permission: 'WRITE'
            },
            {
                Grantee: {
                    Type: 'CanonicalUser',
                    ID: 'your-user-name'
                },
                Permission: 'FULL_CONTROL'
            },
        ],
        Owner: {
            ID: 'your-user-name'
        }
    }
};
s3.putObjectAcl(params, function(err, data) {
    if (err) console.log(err, err.stack);
    else console.log(data);
});
```

Bucket Operation API

Bucket Create

The permitted values of ACL are private, public-read, public-read-write, and authentication-read

```
var params = {
    Bucket: 'test-bucket-xxx',
    ACL: 'public-read'
};
s3.createBucket(params, function(err, data) {
    if (err) console.log(err, err.stack);
    else console.log(data);
});
```

List all the files contained on the bucket.

```
var params = {
    Bucket: 'test-bucket-xxx',
};
s3.listObjects(params, function(err, data) {
    if (err) console.log(err, err.stack);
    else console.log(data);
});
```

Bucket Delete

```
var params = {
    Bucket: 'test-bucket-xxx',
};
s3.deleteBucket(params, function(err, data) {
    if (err) console.log(err, err.stack);
    else console.log(data);
});
```

Get Bucket ACL

```
var params = {
    Bucket: 'test-bucket-xxx',
};
s3.getBucketAcl(params, function(err, data) {
    if (err) console.log(err, err.stack);
    else console.log(data);
});
```

Set Bucket ACL

Using pre-defined ACL

The permitted values of ACL are private, public-read, public-read-write, and authentication-read.

```
var params = {
    Bucket: 'test-bucket-xxx',
    ACL: 'public-read'
};
s3.putBucketAcl(params, function(err, data) {
    if (err) console.log(err, err.stack);
    else console.log(data);
});
```

Using custom ACL

The permitted values of Permission are FULL_CONTROL, WRITE, WRITE_ACP, READ, and READ_ACP

```
var params = {
    Bucket: 'test-bucket-xxx',
    AccessControlPolicy: {
        Grants: [
            {
                Grantee: {
                    Type: 'CanonicalUser',
                    ID: 'user-foo'
                },
                Permission: 'WRITE'
            },
            {
                Grantee: {
                    Type: 'CanonicalUser',
                    ID: 'your-user-name'
                },
                Permission: 'FULL_CONTROL'
            },
        ],
        Owner: {
            ID: 'your-user-name'
        }
    }
};
s3.putBucketAcl(params, function(err, data) {
    if (err) console.log(err, err.stack);
    else console.log(data);
});
```

Service Operation API

Buckets List (List all the Buckets of your service)

```
s3.listBuckets(function(err, data) {
    if (err) console.log(err, err.stack);
    else console.log(data);
});
```

The link of the official AWS SDK for Node.js: aws-sdk-node.js

The link of API documentation: api-reference

Java Demo

Install AWS Java SDK

Using the maven and add the bellowing to the pom.xml

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-java-sdk-bom</artifactId>
            <version>1.12.95</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

<dependencies>
    <dependency>
        <groupId>com.amazonaws</groupId>
        <artifactId>aws-java-sdk-s3</artifactId>
    </dependency>
</dependencies>
```

Initialization & Set the account information and the domain name

```
package com.baishancloud.sdk_test;

import java.io.*;
import java.util.ArrayList;
import java.util.List;
import com.google.gson.*;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.S3ClientOptions;
import com.amazonaws.services.s3.model.*;
import com.amazonaws.ClientConfiguration;

public class App
{
    private static String endPoint = "http://ss.bscstorage.com";
    private static String accessKey = "ziw5dp1alvty9n47qksu"; // please replac
    private static String secretKey = "V+ZTZ5u5wNvXb+KP5g0dMNzhMeWe372/yRKx4hZ

    public static void main( String[] args )
    {
        BasicAWSCredentials awsCreds = new BasicAWSCredentials(
                accessKey, secretKey);

        ClientConfiguration clientconfiguration = new ClientConfiguration();
        clientconfiguration.setSocketTimeout(60 * 60 * 1000); // in millisecon
        clientconfiguration.setConnectionTimeout(60 * 60 * 1000); // in millis

        AmazonS3 s3 = new AmazonS3Client(awsCreds, clientconfiguration);

        s3.setRegion(Region.getRegion(Regions.US_EAST_1));
        s3.setEndpoint(endPoint);
        s3.setS3ClientOptions(S3ClientOptions.builder().setPathStyleAccess(tru
                .disableChunkedEncoding().build());
    }
}
```

File Operation API

File Upload

```
String file_content = "bla bla";
InputStream inputStream = new ByteArrayInputStream(file_content.getBytes());

ObjectMetadata objectMetadata = new ObjectMetadata();
objectMetadata.setContentType("image/jpeg");
objectMetadata.addUserMetadata("key-foo", "value_bar");

PutObjectRequest putObjectrequest = new PutObjectRequest(
        "test-bucket", "test-key", inputStream, objectMetadata);


putObjectrequest.setCannedAcl(CannedAccessControlList.PublicReadWrite);

//AccessControlList accessControlList = new AccessControlList();
//accessControlList.setOwner(new Owner("your_user_id", ""));
//accessControlList.grantPermission(GroupGrantee.AllUsers, Permission.Read);
//accessControlList.grantPermission(GroupGrantee.AllUsers, Permission.ReadAcp)
//accessControlList.grantPermission(GroupGrantee.AuthenticatedUsers, Permissio
//putObjectrequest.setAccessControlList(accessControlList);

PutObjectResult putObjectResult = s3.putObject(putObjectrequest);
```

File Copy

```
CopyObjectResult res = s3.copyObject("srcBucketName", "srcKey", "dstBucketName
```

File Download

```
GetObjectRequest getObjectRequest = new GetObjectRequest("test-bucket", "test-
S3Object s3Object = s3.getObject(getObjectRequest);

System.out.println(s3Object.getObjectMetadata().getUserMetadata().get("key-foo
byte[] buf = new byte[1024];
try {
    int n = s3Object.getObjectContent().read(buf, 0, 1024);
    System.out.println(new String(buf, "UTF-8"));
} catch (IOException e) {
    System.out.println("errr");
}
```

Download to Local File

```
GetObjectRequest getObjectRequest = new GetObjectRequest("test-bucket", "test-
ObjectMetadata meta = s3.getObject(getObjectRequest, new File("/root/test.down
```

Get File URL

Get the pre-signed URL to download the File, and the developer could set an expired time.

```
URL url = s3.generatePresignedUrl("test-bucket", "test-key",
                                  new Date(116, 11, 17)); //2016年12月17日过期
System.out.println(url.toString());
```

File Delete

```
s3.deleteObject("test-bucket", "test_key");
```

Get File ACL

```
AccessControlList accessControlList = s3.getObjectAcl("test-bucket", "test-key
```

Set File ACL

Using pre-defined ACL

```
s3.setObjectAcl(bucketName, "test-key", CannedAccessControlList.AuthenticatedR
```

Using Custom ACL

```
AccessControlList accessControlList = new AccessControlList();
accessControlList.setOwner(new Owner("your_user_id", ""));

accessControlList.grantPermission(GroupGrantee.AuthenticatedUsers, Permission.
accessControlList.grantPermission(GroupGrantee.LogDelivery, Permission.FullCon
accessControlList.grantPermission(new CanonicalGrantee("some_user_id"), Permis
accessControlList.grantPermission(new EmailAddressGrantee("some_email@some.com

s3.setObjectAcl("test-bucket", "test-key", accessControlList);
```

Bucket Operation API

Bucket Create

```
CreateBucketRequest createBucketRequest = new CreateBucketRequest("test-bucket
createBucketRequest.withCannedAcl(CannedAccessControlList.PublicRead);

Bucket bucket = s3.createBucket(createBucketRequest);
```

Bucket File List (List all the files contained on the bucket. The max number of returning files at the same time is 1000)

```
ListObjectsRequest listObjectsRequest = new ListObjectsRequest();
listObjectsRequest.setBucketName("test-bucket");
listObjectsRequest.setMarker("foo"); //设置从哪个key开始列
listObjectsRequest.setPrefix("foo"); //只返回以“foo“为前缀的key
listObjectsRequest.setDelimiter("/"); //对有公共部分的keys进行合并
listObjectsRequest.setMaxKeys(200); //最多返回200个

ObjectListing objectListing = s3.listObjects(listObjectsRequest);
```

List all the files contained on the bucket

```
String marker = "";

ListObjectsRequest listObjectsRequest = new ListObjectsRequest();
listObjectsRequest.setBucketName("test-bucket");
listObjectsRequest.setMarker(marker);

while (true)
{
    ObjectListing objectListing = s3.listObjects(listObjectsRequest);

    List<S3ObjectSummary> contents = new ArrayList<S3ObjectSummary>();
    contents = objectListing.getObjectSummaries();

    if (contents.size() == 0)
    {
        break;
    }

    for (S3ObjectSummary content: contents)
    {
        String key = content.getKey();
        long size = content.getSize();
        System.out.format("key: %s, size: %d\n", key, size);
        listObjectsRequest.setMarker(key);
    }
}
```

Bucket Delete

```
s3.deleteBucket("test-bucket");
```

Get Bucket ACL

```
AccessControlList accessControlList = s3.getBucketAcl("test-bucket");
```

Set Bucket ACL

Using the pre-defined ACL

```
s3.setBucketAcl("test-bucket", CannedAccessControlList.AuthenticatedRead);
```

Using the custom ACL

```
AccessControlList accessControlList = new AccessControlList();
accessControlList.setOwner(new Owner("your_user_id", ""));

accessControlList.grantPermission(GroupGrantee.AuthenticatedUsers, Permission.
accessControlList.grantPermission(GroupGrantee.LogDelivery, Permission.FullCon
accessControlList.grantPermission(new CanonicalGrantee("some_user_id"), Permis
accessControlList.grantPermission(new EmailAddressGrantee("some_email@some.com

s3.setBucketAcl("test-bucket", accessControlList);
```

Service Operation API

Bucket List

```
List<Bucket> allBuckets = new ArrayList<Bucket>();
allBuckets = s3.listBuckets();
```

The link of the official AWS SDK for Java: aws-sdk-java

The link of API documentation: api-reference

Go Demo

Install the AWS Go SDK

```
go get -u github.com/aws/aws-sdk-go/...
```

Initialization & Set the account information and the domain name

```
package main

import "fmt"
import "time"
import "bytes"
import "os"
import "github.com/aws/aws-sdk-go/aws"
import "github.com/aws/aws-sdk-go/service/s3"
import "github.com/aws/aws-sdk-go/service/s3/s3manager"
import "github.com/aws/aws-sdk-go/aws/session"
import "github.com/aws/aws-sdk-go/aws/credentials"

var access_key = "ziw5dp1alvty9n47qksu" // please replace with your access_key
var secret_key = "V+ZTZ5u5wNvXb+KP5g0dMNzhMeWe372/yRKx4hZV" //please replace w
var token = ""
var end_point = "http://ss.bscstorage.com"

func main() {
    credential := credentials.NewStaticCredentials(access_key, secret_key, tok

    config := aws.NewConfig().WithRegion("us-east-1").
        WithEndpoint(end_point).
        WithCredentials(credential).WithS3ForcePathStyle(true)

    sess := session.New(config)
    svc := s3.New(sess)
    uploader := s3manager.NewUploader(sess)
    downloader := s3manager.NewDownloader(sess)
}
```

File Operation API

File Upload

The permitted values of ACL are private, public-read, public-read-write, and authenticated-read.

```
params := &s3.PutObjectInput{
    Bucket: aws.String("test-bucket"),
    Key: aws.String("test-key"),
    ACL: aws.String("public-read"),
    ContentType: aws.String("image/jpeg"), //请替换为合适的文件类型
    Body: bytes.NewReader([]byte("bla bla")),
    Metadata: map[string]*string{
        "key-foo": aws.String("value-bar"),
    },
}

resp, err := svc.PutObject(params)
if err != nil {
    fmt.Println(err.Error())
    return
}
fmt.Println(resp)
```

Using the uploader interface to upload local file

```
f, err := os.Open("/root/test.txt")
if err != nil {
    fmt.Println("open file error")
    return
}

params := &s3manager.UploadInput{
    Bucket: aws.String("test-bucket"),
    Key: aws.String("test-key"),
    Body: f,
}

result, err := uploader.Upload(params)
if err != nil {
    fmt.Println("upload file error")
    return
}
fmt.Printf("file uploaded to: %s\n", result.Location)
```

File Download

```
params := &s3.GetObjectInput{
    Bucket: aws.String("test-bucket"),
    Key: aws.String("test-key"),
}

resp, err := svc.GetObject(params)
if err != nil {
    fmt.Println(err.Error())
    return
}
buf := new(bytes.Buffer)
buf.ReadFrom(resp.Body)
fmt.Println(buf.String())
```

Using the downloader interface to download the file

```
f, err := os.Create("/root/test.txt.download")
if err != nil {
    fmt.Println("create file error")
    return
}

params := &s3.GetObjectInput{
    Bucket: aws.String("test-bucket"),
    Key: aws.String("test-key"),
}

n, err := downloader.Download(f, params)
if err != nil {
    fmt.Println("download file error")
    return
}
fmt.Printf("file download %d bytes\n", n)
```

Get File URL

Get the pre-signed URL to download the File, and the developer could set an expired time.

```
params := &s3.GetObjectInput{
    Bucket: aws.String("test-bucket"),
    Key: aws.String("test-key"),
}
req, _ := svc.GetObjectRequest(params)
url, _ := req.Presign(300 * time.Second) //300秒后过期
fmt.Println(url)
```

File Delete

```
params := &s3.DeleteObjectInput{
    Bucket: aws.String("test-bucket"),
    Key: aws.String("test-key"),
}
resp, err := svc.DeleteObject(params)
if err != nil {
    fmt.Println(err.Error())
    return
}
fmt.Println(resp)
```

Get File ACL

```
params := &s3.GetObjectAclInput{
    Bucket: aws.String("test-bucket"),
    Key: aws.String("test-key"),
}

resp, err := svc.GetObjectAcl(params)
if err != nil {
    fmt.Println(err.Error())
    return
}
fmt.Println(resp)
```

Set File ACL

Using pre-defined ACL

支持的预定义ACL有：'private', 'public-read', 'public-read-write' 和 'authenticated-read'

```
params := &s3.PutObjectAclInput{
    Bucket: aws.String("test-bucket"),
    Key: aws.String("test-key"),
    ACL: aws.String("private"),
}

resp, err := svc.PutObjectAcl(params)
if err != nil {
    fmt.Println(err.Error())
    return
}
fmt.Println(resp)
```

Using Grant header to Set ACL

```
params := &s3.PutObjectAclInput{
    Bucket: aws.String("test-bucket"),
    Key: aws.String("test-key"),

    GrantFullControl: aws.String("id=your_user_id"),
    GrantRead: aws.String("uri=\"http://acs.amazonaws.com/groups/global/AllUse
    GrantReadACP: aws.String("id=some_user_id, id=another_user_id"),
    GrantWrite: aws.String("uri=\"http://acs.amazonaws.com/groups/global/Authe
    GrantWriteACP: aws.String("emailAddress=\"some_email@some.com\", id=some_u
}

resp, err := svc.PutObjectAcl(params)
if err != nil {
    fmt.Println(err.Error())
    return


}
fmt.Println(resp)
```

Using custom ACL

The permitted values of Permission are FULL_CONTROL, WRITE, WRITE_ACP, READ, and READ_ACP

```
params := &s3.PutObjectAclInput{
    Bucket: aws.String("test-bucket"),
    Key: aws.String("test-key"),

    AccessControlPolicy: &s3.AccessControlPolicy{
        Grants: []*s3.Grant{
            {
                Grantee: &s3.Grantee{
                    Type: aws.String("CanonicalUser"),
                    DisplayName: aws.String(""),
                    ID: aws.String("some_user_id"),
                },
                Permission: aws.String("FULL_CONTROL"),
            },
            {
                Grantee: &s3.Grantee{
                    Type: aws.String("Group"),
                    URI: aws.String("http://acs.amazonaws.com/groups/global/Al
                },
                Permission: aws.String("READ"),
            },
            {
                Grantee: &s3.Grantee{
                    Type: aws.String("AmazonCustomerByEmail"),
                    DisplayName: aws.String(""),
                    EmailAddress: aws.String("some_email@some.com"),
                },
                Permission: aws.String("READ"),
            },
        },
        Owner: &s3.Owner{
            DisplayName: aws.String(""),
            ID: aws.String("your_user_id"),
        },
    },
}

resp, err := svc.PutObjectAcl(params)
if err != nil {
    fmt.Println(err.Error())
    return
}
fmt.Println(resp)
```

Bucket Operation API

Bucket Create

The permitted values of ACL are private, public-read, public-read-write, and authentication-read

```
params := &s3.CreateBucketInput{
    Bucket: aws.String("test-bucket"),
    ACL: aws.String("public-read"),
}

resp, err := svc.CreateBucket(params)
if err != nil {
    fmt.Println(err.Error())
    return
}
fmt.Println(resp)
```

List all the files contained on the bucket (every time it can return up to 1000 files)

```
params := &s3.ListObjectsInput{
    Bucket: aws.String("test-bucket"),
    Marker: aws.String("foo"), //设置从哪个key开始列
    Prefix: aws.String("foo"), //只返回以“foo“为前缀的key
    Delimiter: aws.String("/"), //对含有公共部分的keys进行合并
    MaxKeys: aws.Int64(200), //最多返回200个
}

resp, err := svc.ListObjects(params)
if err != nil {
    fmt.Println(err.Error())
    return
}
fmt.Println(resp)
```

List all the files on the bucket

```
marker := ""

for {
    params := &s3.ListObjectsInput{
        Bucket: aws.String("test-bucket"),
        Marker: aws.String(marker),
    }

    resp, err := svc.ListObjects(params)
    if err != nil {
        fmt.Println(err.Error())
        return
    }

    if len(resp.Contents) == 0 {
        break;
    }

    for _, content := range resp.Contents {
        fmt.Printf("key:%s, size:%d\n", *content.Key, *content.Size)
        marker = *content.Key
    }
}
```

Bucket Delete

```
params := &s3.DeleteBucketInput{
    Bucket: aws.String("test-bucket"),
}
resp, err := svc.DeleteBucket(params)
if err != nil {
    fmt.Println(err.Error())
    return
}
fmt.Println(resp)
```

Get Bucket ACL

```
params := &s3.GetBucketAclInput{
    Bucket: aws.String("test-bucket"),
}
resp, err := svc.GetBucketAcl(params)
if err != nil {
    fmt.Println(err.Error())
    return
}
fmt.Println(resp)
```

Set Bucket ACL

Using the pre-defined ACL

The permitted values of ACL are private, public-read, public-read-write, and authentication-read.

```
params := &s3.PutBucketAclInput{
    Bucket: aws.String("test-bucket"),
    ACL: aws.String("public-read-write"),
}

resp, err := svc.PutBucketAcl(params)
if err != nil {
    fmt.Println(err.Error())
    return
}
fmt.Println(resp)
```

Set ACL by using Grant header

```
params := &s3.PutBucketAclInput{
    Bucket: aws.String("test-bucket"),

    GrantFullControl: aws.String("id=your_user_id"),
    GrantRead: aws.String("uri=\"http://acs.amazonaws.com/groups/global/AllUse
    GrantReadACP: aws.String("id=some_user_id, id=another_user_id"),
    GrantWrite: aws.String("uri=\"http://acs.amazonaws.com/groups/global/Authe
    GrantWriteACP: aws.String("emailAddress=\"some_email@some.com\", id=some_u
}

resp, err := svc.PutBucketAcl(params)
if err != nil {
    fmt.Println(err.Error())
    return
}
fmt.Println(resp)
```

Using Custom ACL

The permitted values of Permission are FULL_CONTROL, WRITE, WRITE_ACP, READ, and READ_ACP

```
params := &s3.PutBucketAclInput{
    Bucket: aws.String("test-bucket"),

    AccessControlPolicy: &s3.AccessControlPolicy{
        Grants: []*s3.Grant{
            {
                Grantee: &s3.Grantee{
                    Type: aws.String("CanonicalUser"),
                    DisplayName: aws.String(""),
                    ID: aws.String("some_user_id"),
                },
                Permission: aws.String("FULL_CONTROL"),
            },
            {
                Grantee: &s3.Grantee{
                    Type: aws.String("Group"),
                    URI: aws.String("http://acs.amazonaws.com/groups/global/Al
                },
                Permission: aws.String("READ"),
            },
            {
                Grantee: &s3.Grantee{
                    Type: aws.String("AmazonCustomerByEmail"),
                    DisplayName: aws.String(""),
                    EmailAddress: aws.String("some_email@some.com"),
                },
                Permission: aws.String("READ"),
            },
        },
        Owner: &s3.Owner{
            DisplayName: aws.String(""),
            ID: aws.String("your_user_id"),
        },
    },
}

resp, err := svc.PutBucketAcl(params)
if err != nil {
    fmt.Println(err.Error())
    return
}
fmt.Println(resp)
```

Service Operation API

List all the Buckets

```
var params *s3.ListBucketsInput
resp, err := svc.ListBuckets(params)
if err != nil {
    fmt.Println(err.Error())
    return
}
fmt.Println(resp)
```

The link of the official AWS SDK for Go: aws-sdk-go

The link of API documentation: api-reference

Dotnet Demo

Install AWS Dotnet SDK

```
dotnet add package AWSSDK.S3
```

Initialization & Set the account information and the domain name

```
using System;
using System.IO;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;

namespace dnet
{
    class Program
    {
        private static IAmazonS3 client;

        public static void Main()
        {
            # Please fill in it with your own access key
            string accessKey = "accessKey";

            # Please fill in it with your own secret key
            string secretKey = "secretKey";

            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://ss.bscstorage.com";
            client = new AmazonS3Client(accessKey, secretKey, config);
        }
    }
}
```

File Operation API

File Upload

```
static async Task WritingAnObjectAsync()
{
    try
    {
        // 1. Put object-specify only key name for the new object.
        var putRequest1 = new PutObjectRequest
        {
            BucketName = "test-bucket-xxx",
            Key = "test-key-xxx",
            ContentBody = "sample text"
        };

        PutObjectResponse response1 = await client.PutObjectAsync(putRequest1)

        // 2. Put the object-set ContentType and add metadata.
        var putRequest2 = new PutObjectRequest
        {
            BucketName = "test-bucket-xxx",
            Key = "test-key-xxx",
            FilePath = "/root/test.txt",
            ContentType = "text/plain"
        };
        putRequest2.Metadata.Add("x-amz-meta-title", "someTitle");
        PutObjectResponse response2 = await client.PutObjectAsync(putRequest2)
    }
    catch (AmazonS3Exception e)
    {
        Console.WriteLine(
                "Error encountered ***. Message:'{0}' when writing an object"
                , e.Message);
    }
    catch (Exception e)
    {
        Console.WriteLine(
            "Unknown encountered on server. Message:'{0}' when writing an obje
            , e.Message);
    }
}
```

File Download

```
static async Task ReadObjectDataAsync()
{
    string responseBody = "";
    try
    {
        GetObjectRequest request = new GetObjectRequest
        {
            BucketName = bucketName,
            Key = keyName
        };
        using (GetObjectResponse response = await client.GetObjectAsync(reques
        using (Stream responseStream = response.ResponseStream)
        using (StreamReader reader = new StreamReader(responseStream))
        {
            string title = response.Metadata["x-amz-meta-title"]; // Assume yo
            string contentType = response.Headers["Content-Type"];
            Console.WriteLine("Object metadata, Title: {0}", title);
            Console.WriteLine("Content type: {0}", contentType);

            responseBody = reader.ReadToEnd(); // Now you process the response
        }
    }
    catch (AmazonS3Exception e)
    {
        Console.WriteLine("Error encountered ***. Message:'{0}' when writing a
    }
    catch (Exception e)
    {
        Console.WriteLine("Unknown encountered on server. Message:'{0}' when w
    }
}
```

File Delete

```
private static async Task DeleteObjectAsync()
{
    try
    {
        // Delete the object
        DeleteObjectRequest request = new DeleteObjectRequest
        {
            BucketName = bucketName,
            Key = keyName,
        };
        Console.WriteLine("Deleting an object");
        await client.DeleteObjectAsync(request);
    }
    catch (AmazonS3Exception e)
    {
        Console.WriteLine("Error encountered on server. Message:'{0}' when wri
    }
    catch (Exception e)
    {
        Console.WriteLine("Unknown encountered on server. Message:'{0}' when w
    }
}
```

Get File ACL

```
private static async Task TestObjectACLAsync()
{
    try
    {
        // Retrieve the ACL for the object.
        GetACLResponse aclResponse = await client.GetACLAsync(new GetACLReques
        {
            BucketName = bucketName,
            Key = keyName
        });

        S3AccessControlList acl = aclResponse.AccessControlList;

        // Retrieve the owner (we use this to re-add permissions after we clea
        Owner owner = acl.Owner;

        // Clear existing grants.
        acl.Grants.Clear();

        // Add a grant to reset the owner's full permission (the previous clea
        S3Grant fullControlGrant = new S3Grant
        {
            Grantee = new S3Grantee { CanonicalUser = owner.Id },
            Permission = S3Permission.FULL_CONTROL
        };

        // Describe the grant for the permission using an email address.
        S3Grant grantUsingEmail = new S3Grant
        {
            Grantee = new S3Grantee { EmailAddress = emailAddress },
            Permission = S3Permission.WRITE_ACP
        };
        acl.Grants.AddRange(new List<S3Grant> { fullControlGrant, grantUsingEm

        // Set a new ACL.
        PutACLResponse response = await client.PutACLAsync(new PutACLRequest
        {
            BucketName = bucketName,
            Key = keyName,
            AccessControlList = acl
        });
    }
    catch (AmazonS3Exception amazonS3Exception)
    {
        Console.WriteLine("An AmazonS3Exception was thrown. Exception: " + ama
    }
    catch (Exception e)
    {
        Console.WriteLine("Exception: " + e.ToString());
    }
}
```

Bucket Operation API

Bucket Create

```
static async Task CreateBucketAsync()
{
    try
    {
        PutBucketResponse response = await client.PutBucketAsync("test-bucket-
    }
    catch (AmazonS3Exception e)
    {
        Console.WriteLine("Error encountered on server. Message:'{0}' when wri
    }
    catch (Exception e)
    {
        Console.WriteLine("Unknown encountered on server. Message:'{0}' when w
    }
}
```

List all the Files on the Bucket

```
static async Task ListObjectsAsync()
{
    try
    {
        // List all objects
        ListObjectsRequest listRequest = new ListObjectsRequest
        {
            BucketName = "test-bucket-xxx",
            MaxKeys = 10,
        };

        ListObjectsResponse listResponse;
        do
        {
            // Get a list of objects
            listResponse = client.ListObjectsAsync(listRequest).Result;
            foreach (S3Object obj in listResponse.S3Objects)
            {
                Console.WriteLine("key = {0} size = {1}", obj.Key, obj.Size);
            }

            // Set the marker property
            listRequest.Marker = listResponse.NextMarker;
        } while (listResponse.IsTruncated);

    }
    catch (AmazonS3Exception e)
    {
        Console.WriteLine("Error encountered on server. Message:'{0}' when wri
    }
    catch (Exception e)
    {
        Console.WriteLine("Unknown encountered on server. Message:'{0}' when w
    }
}
```

Bucket Delete

```
static async Task DeleteBucketAsync()
{
    try
    {
        var response = await client.DeleteBucketAsync("test-bucket-xxx");
    }
    catch (AmazonS3Exception e)
    {
        Console.WriteLine("Error encountered on server. Message:'{0}' when wri
    }
    catch (Exception e)
    {
        Console.WriteLine("Unknown encountered on server. Message:'{0}' when w
    }
}
```

Get Bucket ACL

```
static async Task GetBucketACLAsync(string bucketName)
{
    try
    {
        GetACLResponse response = await client.GetACLAsync(new GetACLRequest
        {
            BucketName = "test-bucket-xxx"
        });
        S3AccessControlList accessControlList = response.AccessControlList;
        Console.WriteLine("ID: {0}", accessControlList.Owner.Id);
        foreach(var g in accessControlList.Grants)
        {
            Console.WriteLine("Permission: {0}, {1}", g.Permission.HeaderName,
            Console.WriteLine("Grantee: {0}", g.Grantee.CanonicalUser);
        }
    }
    catch (AmazonS3Exception e)
    {
        Console.WriteLine("Error encountered on server. Message:'{0}' when wri
    }
    catch (Exception e)
    {
        Console.WriteLine("Unknown encountered on server. Message:'{0}' when w


    }
}
```

Set Bucket ACL

```
// Set Canned ACL (PublicRead)
client.PutACL(new PutACLRequest
{
    BucketName = "SampleBucket",
    CannedACL = S3CannedACL.PublicRead
});

// Set Canned ACL (PublicRead)
client.PutACL(new PutACLRequest
{
    BucketName = "SampleBucket",
    CannedACL = S3CannedACL.Private
});
```

List all the Buckets

```
static async Task ListBucketsAsync(string bucketName)
{
    var response = client.ListBucketsAsync().Result;
    foreach (var bucket in response.Buckets)
    {
        Console.WriteLine(bucket.BucketName);
    }
}
```

The link of the official AWS SDK for Dotnet: aws-sdk-dotnet

The link of API documentation: api-reference

Constraint and limitation

- Coding restriction
- Bucket name, object key, meta, ACL, etc. only supports UTF-8 encoding
- The URL to access the resource should be encoded by rawurlencode
- Object key needs to be encoded by rawurlencode except "/"
- Naming rule

The name of bucket and object in API shall comply with the following rules:

Bucket name:

- It is unique in cloud storage
- It is composed of the lowercase letter, the number and the character "-" and the length should be from 6 to 63 digits;
- Cannot start with the number and the character '-'；
- Cannot start or end with '-'；
- Or use the domain name, such as xxx.foo.com.cn, which is convenient for binding your domain name

Object name:

- The length of the key cannot exceed 512 bytes；
- The key excepts "/" should be encoded by rawurlencode

Signature algorithm

- If the HTTP request does not carry the identity information (AccessKey), the request will be changed to an anonymous request, which will be considered by an anonymous user.
- If the identity information is carried in the HTTP request (AccessKey), the request will be considered by the user who is corresponding to the access Key. Since the AccessKey can be obtained by others, in order to prevent others from using your accessKey to access the service, your signature must also be carried in the request. After applying for an account, you will get the AccessKey and Secret key, which need to be kept confidential. The signature will be calculated by the HTTP request Information and your secret key so that others do not know your secret key, they will not be able to calculate the correct signature.
- The identity information and signature can be put in the request header （Authorization）or in the request parameters.
- The signature method is compatible with Amazon S3 and supports signature version 2 and signature version 4.

Add Signature

> Since the process of calculating the signature is cumbersome and error-prone, it is not recommended to calculate the signature by yourself. It is recommended to use the SDKs. The SDK can automatically calculate the signature.

Add version 2 Signature

Carrying method of identity information and signature:

Request header via Authorization:

Request header format:

```
Authorization: AWS AWSAccessKeyId:Signature
```

- AWSAccessKeyId: your AccessKey
- Signature: calculated signature

```
Authorization: AWS ziw5dp1alvty9n47qksu:frJIUN8DYpKDtOLCwo//yllqDzg=
```

Request parameters:

You need to include the following three parameters in the request:

- Awsaccesskeyid: To specify your AccessKey

- Signature: calculated signature
- Expires: Specifies the expiration time of the signature

Example:

```
GET /yourbucket/yourkey?AWSAccessKeyId=ziw5dp1alvty9n47qksu&Expires=1141889120
```

Signature calculation method

```
Signature = Base64( HMAC-SHA1( YourSecretKey, UTF-8-Encoding-Of( StringToSign

StringToSign = HTTP-Verb + "\n" +

Content-MD5 + "\n" +

Content-Type + "\n" +

Date|Expires + "\n" +

CanonicalizedAmzHeaders +

CanonicalizedResource
```

- Yoursecretkey: To fill in it with your Secretkey
- HTTP-Verb: The request method, such as PUT, GET, DELETE, POST
- Content-MD5: The request header of the content-MD5. If there is no such header, please fill in it with an empty string
- Content-Type: The request header of the content-Type. If there is no such header, please fill in it with an empty string
- Date|Expires: If the Authorization header is used to be as the content of the Date header. please fill in it with an empty string when there is no the Date Header. If the request parameter is used to carry the signature information, it will be the content of the Expires.
- CanonicalizedAmzHeaders: The string composed of all headers starting with x-amz - in the request. If there is no such header, please fill in it with an empty string.
- CanonicalizedResource: The Resource which is corresponding to the request.

Example of calculating the Canonicalizedamzheaders:

Original request header:

```
Date: Tue, 27 Mar 2007 19:36:42 +0000
X-Amz-b:   Bar
x-amz-a: foob
x-Amz-a:   fooa
Host: johnsmith.s3.amazonaws.com

The corresponding CanonicalizedAmzHeaders is:
x-amz-a:fooa,foob
x-amz-b:Bar
```

> Note:
>
> 1. All the request header names are converted to lowercase and sorted by the converted header name. 2. If the duplicate headers appeared, please merge them, such as separating the value by commas and sorting them. 3. Remove the spaces before and after the value.

Example of calculating Canonicalizedresource:

```
GET /?foo=bar

GET /yourbucket/yourkey?foo=bar

GET /yourbucket/yourkey?acl&foo=bar

The corresponding nicalizedresources are:

/

/yourbucket/yourkey

/yourbucket/yourkey?acl
```

If you want to know the completed signature calculation process, please refer to the following links:

- http://docs.aws.amazon.com/AmazonS3/latest/dev/RESTAuthentication.html
- http://docs.aws.amazon.com/general/latest/gr/signature-version-2.html

Add version 4 Signature:

Carrying method of identity information and signature:

Through the Authorization header:

example:

```
Authorization:AWS4HMACSHA256Credential=ziw5dp1alvty9n47qksu/20160830/useast-1/
```

- Credential consists of AccessKey, the requested date, region, service name, and aws4_ Request, which is separated by slashes
- SignedHeaders: Indicates which header participates in the calculation of the signature. Headers not included here will not affect the generation of the signature
- Signature: Calculated signature

Through request parameters:

The following parameters need to be added to the request:

- X-Amz-Algorithm: The hash algorithm used to calculate the signature which is AWS4-HMAC-SHA256
- X-Amz-Credential: It contains the information of AccessKey, date, region, and service name

- X-Amz-Date: Requested time
- X-Amz-Expires: Specify the expired time of the signature
- X-Amz-SignedHeaders: The header which is used to calculate the signature
- X-Amz-Signature: Calculated signature

Example:

```
GET /yourbucket/test.mp4??X-Amz-Algorithm=AWS4-HMAC-SHA256&&X-Amz-Credential=z
```

Signature calculation method:

Calculate CanonicalRequest

```
CanonicalRequest =

HTTPRequestMethod + '\n' +

CanonicalURI + '\n' +

CanonicalQueryString + '\n' +

CanonicalHeaders + '\n' +

SignedHeaders + '\n' +

HexEncode(Hash(RequestPayload))
```

- HTTPRequestMethod: PUT, GET, DELETE, POST
- CanonicalURI: Requested URI
- CanonicalQueryString: The string formed after sorting the request parameters
- CanonicalHeaders: The string formed after sorting the header needs to be added to the signature calculation
- SignedHeaders：A list of names added to the header of the signature calculation, separated by commas
- HexEncode(Hash(RequestPayload))： The hexadecimal encoding of the hash of the request body. If the signature is carried through the request parameters, the string unsigned-payload should be used instead

Calculate StringToSign

```
StringToSign  =

Algorithm + '\n' +

RequestDate + '\n' +

CredentialScope + '\n' +

HashedCanonicalRequest
```

- Algorithm: AWS4-HMAC-SHA256
- RequestDate: ISO8601 basic Request time in format, e.g:20160830T123600Z

- CredentialScope: A string consisting of date, region, service name, etc., such as 20160830/us-east-1/s3/aws4_request
- HashedCanonicalRequest: Hex(SHA256Hash(CanonicalRequest)), That is, the hexadecimal encoding of the hash of canonicalrequest

Calculation signing key

```
kSecret = YourSecretKey
kDate = HMAC("AWS4" + kSecret, Date)
kRegion = HMAC(kDate, Region)
kService = HMAC(kRegion, Service)
kSigning = HMAC(kService, "aws4\_request")
```

- YourSecretKey: your SecretKey
- Date：The 8-digit date should be the same as the date part in credentail
- Region：It should be the same as the region section in credential
- Service: It should be the same as the service name part in credential

Calculate signature

```
signature = HexEncode(HMAC-SHA256(kSigning, StringToSign))
```

If you want to know the complete and detailed signature calculation process, please refer to the following connection:

- http://docs.aws.amazon.com/AmazonS3/latest/API/sig-v4-authenticating-requests.html
- http://docs.aws.amazon.com/general/latest/gr/signature-version-4.html

ACL(Access Control List)

Access control lists (ACL) enable you to manage access to buckets and objects. Each bucket and object has an additional ACL sub resource. It defines which users or groups will be granted access rights. After receiving a request for a resource (bucket or object), S2 will check the corresponding ACL to verify whether the requester has the required access rights.

When creating a bucket or object, S2 will create a default ACL to grant the resource owner full control over the resource, as shown in the following example bucket ACL.

```
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Owner>
        <ID>*** Owner-Canonical-User-ID ***</ID>
        <DisplayName>owner-display-name</DisplayName>
    </Owner>
    <AccessControlList>
        <Grant>
            <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi
                <ID>*** Owner-Canonical-User-ID ***</ID>
                <DisplayName>display-name</DisplayName>
            </Grantee>
            <Permission>FULL_CONTROL</Permission>
        </Grant>
    </AccessControlList>

</AccessControlPolicy>
```

The example ACL contains an owner element that represents the user name of the resource owner. The grant element represents the authorized person (user or predefined group) and the permissions granted. You can grant permissions by adding the grant element.

Grantee

The authorized person can be an S2 user or a predefined S2 group. You can grant permissions to a user by email address or user name. However, if you set an email address in the authorization request, S2 will find the user name corresponding to the email and add it to the ACL. The generated ACL will always contain the S2 user's user name, not the user's email address.

S2 predefined groups

S2 has a series of predefined groups. When granting user access to a group, you can specify a URI instead of a user name. We provide the following predefined groups:

- All Users group, identified by http://acs.amazonaws.com/groups/global/AllUsers. The access rights of this group allow anyone to access the resource. Requests can be signed (authenticated) or unsigned (anonymous).
- Log Delivery group, identified by http://acs.amazonaws.com/groups/s3/LogDelivery, the WRITE permission on the storage bucket allows this group to write server access logs to the storage bucket.

Specify the permissions of the authorized person

The following table lists the permission sets supported by S2 in ACL and its meaning in the context of operation resources.

Permission	When authorizing on the storage bucket	When authorizing on an object
READ	The authorizer is allowed to list all objects in the bucket	Allow the authorized person to read the data and meta information of the object
WRITE	Allows authorized persons to create, overwrite, and delete objects in buckets	Not available
READ_ACP	Allow authorized persons to read buckets ACL	允许被授权者读取对象的ACL
WRITE_ACP	Allows the authorized person to modify the ACL of the bucket	Allows the authorized person to modify the ACL of an object
FULL_CONTROL	Allow the authorized person to READ、WRITE、READ_ACP and WRITE_ACPpermissions	Allow the authorized person on the objectREAD、READ_ACP 和 WRITE_ACP permissions

Canned ACL

S2 supports a series of predefined authorizations, called standard ACLs. each standard ACL has a predefined set of authorized persons and permissions. The following table lists a set of standard ACLs and the associated predefined authorizations.

Canned ACL	Applicable for	Corresponding ACL permissions
private	Buckets and objects	The owner will get FULL_CONTROL。Others do not have access (default).
public-read	Buckets and objects	The owner will get FULL_CONTROL。Anonymous users have READpermissions
public-read-write	Buckets and objects	The owner will get FULL_CONTROL。AllUsers have READ, WRITEpermissions,Object no WRITE permissions
authenticated-read	Buckets and objects	The owner will get FULL_CONTROL。AuthenticatedUsers have READ

How to specify ACL

The S2 API allows you to set ACL when creating buckets or objects. S2 also provides an API to set ACL on existing buckets or objects. These APIs provide you with the following methods to set ACL:

- Set ACL with request header - When sending a request to create a resource (bucket or object), you can use the x-amz-acl header to set the Canned acl, or the x-amz-grant-* header to set the permissions for a user or group.
- Set ACL using ACL API - When you send a request to set an ACL on an existing resource, you can set the ACL in the request header or body, such as the put_object_acl API

Attention:

- x-amz-acl and x-amz-grant-* cannot be specified at the same time, both of them return 400 errors
- If x-amz-acl or x-amz-grant-* request header is specified in the put_object_acl API, the ACL in the body is ignored.

x-amz-grant-* request header type

x-amz-grant- *the value of the request header is in the form of a type = value , each tpye = value is separated by commas. Multiple x-amz-grant-* request headers can be specified in the same request. The allowed types are:

- emailAddress, User's email address
- id, user name
- uri, Uri for predefined user groups

Name	Description
x-amz-grant-read	Permission to allow the authorizer to READ
x-amz-grant-write	Permission to allow the authorizer to WRITE
x-amz-grant-read-acp	Permission to allow the authorizer to READ_ACP
x-amz-grant-write-acp	Permission to allow the authorizer to WRITE_ACP
x-amz-grant-full-control	Permission to allow the authorizer to READ, WRITE, READ_ACP, WRITE_ACP

Example:

```
x-amz-grant-read: emailAddress="xyz@amazon.com", emailAddress="abc@amazon.com"
```

Service Operations

GET Service (List Buckets)

- Description: get the list of all buckets under the current owner.
- Request Syntax

```
GET /?formatter=json HTTP/1.1
Host: ss.bscstorage.com
Date: <date>
Authorization: <authorization string> # Please refer to 'Signature algorithm'
```

or

```
GET /<Your-Bucket-Name>/?formatter=json HTTP/1.1
Host: ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #Please refer to signature algorithm
```

- Request Syntax（HTTP Body）：

```
{
    "Owner": {
        "DisplayName": "",
        "ID": "Baishan0000001234567890"
    },

    "Buckets": {
        "Bucket": [
            {
                "CreationDate": "Fri, 21 Mar 2014 01:13:42 UTC",
                "Name": "bucket_name_0"
            },
            {
                "CreationDate": "Fri, 12 Mar 2013 02:25:22 UTC",
                "Name": "bucket_name_1"
            },

            ...
        ]
    }
}
```

- Return value Description:

Name	Description
Owner	owner
DisplayName	Display name of the owner
ID	Owner's UserId
Buckets	Containers for multiple buckets
Bucket	Bucket container for information
CreationDate	Current bucket creation date
Name	Bucket name

- Request example:

```
curl -v -H "Date: Sat, 20 Nov 2286 17:46:39 GMT" -H "Authorization: Baishan <a
```

or

```
curl -v "http://ss.bscstorage.com/?KID=baishan,<access_key>&Expires=1398873316
```

- Request example：

```
HTTP/1.1 200 OK
Server: openresty/1.9.7.4
Date: Mon, 08 Aug 2016 04:04:52 GMT
Content-Type: application/json
Connection: keep-alive
Content-Length: 155
x-amz-s2-requester: your user id
x-amz-request-id: 000011e5-1608-0812-0452-00163e0069ec

{
    "Owner": {
        "DisplayName": "",
        "ID": "Baishan0000001234567890"
    },
    "Buckets": {
        "Bucket": [
            {
                "CreationDate": "Mon, 08 Aug 2016 03:15:40 UTC",
                "Name": "bucket_name_0"
            },
            {
                "CreationDate": "Mon, 08 Aug 2016 03:15:40 UTC",
                "Name": "bucket_name_1"
            },

         ...
        ]
    }
}
```

Bucket Operations

GET Bucket (List Objects)

- Description: get all objects under the bucket.
- Request Syntax:

```
GET /?formatter=json HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #Please refer to signature algorithm
```

or

```
GET /<Your-Bucket-Name>/?formatter=json HTTP/1.1
Host: ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #Please refer to signature algorithm
```

- Request parameters:

Parameter	Description	Required
delimiter	Folded display characters. Usually use：‘/’	No
marker	The initial position of the Key, the system will list the value larger than the Key, usually used as a 'paging' scenario	No
max-keys	The default maximum number of Key value is 400	No
prefix	List the Keys that start with the specified character	No

Example of response format（HTTP Body）：

```
{

    Delimiter: "/",

    Prefix: "html/",

    CommonPrefixes: [

        {
            Prefix: "html/assets/"
        },

        {
            Prefix: "html/attributions/"
        },

        {
            Prefix: "html/documentation/"
        },

        ...
    ],

    Marker: null,

    ContentsQuantity: 5,

    CommonPrefixesQuantity: 3,

    NextMarker: null,

    IsTruncated: false,

    Contents: [

        {
            SHA1: "9fc710aa89efbe42020b0310d16a07449bf06131",
            Name: "html/coming-soon.html",
            Expiration-Time: null,
            Last-Modified: "Fri, 21 Mar 2014 01:50:46 UTC",
            Owner: "Baishan0000000000000001",
            MD5: "934d922cac80449ee361cefeb3276b3e",
            Content-Type: "text/html",
            Size: 8781
        },

        {
            SHA1: "a9625a128263f05e331f6d78add9bd15911c3565",
            Name: "html/ebook.html",
            Expiration-Time: null,
            Last-Modified: "Fri, 21 Mar 2014 01:50:47 UTC",
            Owner: "Baishan0000000000000001",
            MD5: "cb7ed943ead4aeb513aa8c0b76865a8b",
            Content-Type: "text/html",
            Size: 18734
        },

        ...
    ]
}
```

- Return value Description:

Name	Description
Contents	Metadata array of object
CommonPrefixes	After folding the prefix, the next level is the prefix array
Delimiter	Currently used collapse character
Prefix	Currently used prefix
Marker	Currently used marker
ContentsQuantity	Number of elements in contents
CommonPrefixesQuantity	Number of elements in Common Prefixes
NextMarker	Marker on next page
IsTruncated	Is there another page
SHA1	SHA1 value of file content
Name	Key (file name) of object
Last-Modified	Last modification time of object
Owner	Owner of the object
MD5	MD5 value of file content
Content-Type	MIME-type of the file
Size	File size (bytes)

- Example:

If there are some files as follows under a bucket (for convenience, they are not displayed in the JSON format, only show some useful information, the same below):

```
join/mailaddresss.txt
join/mycodelist.txt
join/personalfiles/connects.docx
join/personalfiles/myphoto.jpg
join/readme.txt
join/userlist.txt
join/zero.txt
mary/personalfiles/mary.jpg
mary/readme.txt
sai/readme.txt
```

Use prefix to specify files starting with join /

```
GET /?prefix=join/&formatter=json HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #Please refer to signature algorithm
```

return:

```
Contents:
    join/mailaddresss.txt
    join/mycodelist.txt
    join/personalfiles/connects.docx
    join/personalfiles/myphoto.jpg
    join/readme.txt
    join/userlist.txt
    join/zero.txt
```

Use the delimiter to specify that the collapse method is '/'：

```
GET /?delimiter=/&formatter=json HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #Please refer to signature algorithm
```

return：

```
Contents:

CommonPrefix:
    join
    mary
    sai
```

Use prefix to specify the file starting with join /, and use delimiter to specify the folding method as '/'：

```
GET /?prefix=join/&delimiter=/&formatter=json HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #请参照《签名算法》
```

return：

```
Contents:
    join/mailaddresss.txt
    join/mycodelist.txt
    join/readme.txt
    join/userlist.txt
    join/zero.txt
CommonPrefix:
    join/personalfiles/
```

Use max-keys to limit the maximum list length:

```
GET /?prefix=join/&delimiter=/&max-keys=4&formatter=json HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #请参照《签名算法》
```

return：

```
IsTruncated : true
Next-Marker : join/userlist.txt
Contents:
    join/mailaddresss.txt
    join/mycodelist.txt
    join/readme.txt
CommonPrefix:
    join/personalfiles/
```

Use marker to continue to get the subsequent results of the previous column operations：

```
GET /?prefix=join/&delimiter=/&max-keys=4&marker=join/userlist.txt&formatter=j
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #请参照《签名算法》
```

return：

```
IsTruncated : false
Contents:
    join/userlist.txt
    join/zero.txt
```

PUT Bucket

- Description: create a bucket.
- Request Syntax

```
PUT /?formatter=json HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #Please refer to signature algorithm
x-amz-acl: <Canned-ACL> #Please refer to[《ACL》](path: ../acl/acl.md)
```

or

```
PUT /<Your-Bucket-Name>/?formatter=json HTTP/1.1
Host: ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #Please refer to signature algorithm
x-amz-acl: <Canned-ACL> #Please refer to[《ACL》](path: ../acl/acl.md)
```

- Request Header（请求头）：

Name	Description	Required
x-amz-acl	Set an ACL while creating a bucket. Please refer to ACL	No

- Response (no HTTP body)：

```
HTTP/1.1 200 OK
Date: Tue, 08 Apr 2014 02:59:47 GMT
Content-Length: 0
Connection: keep-alive
X-RequestId: 00078d50-1404-0810-5947-782bcb10b128
X-Requester: Your UserId
```

- Request example:

```
curl -v -X PUT -H "Date: Sat, 20 Nov 2286 17:46:39 GMT" -H "Authorization: Bai
```

or

```
curl -v -X PUT "http://<Your-Bucket-Name>.ss.bscstorage.com/?KID=baishan,<acce
```

or

```
curl -v -X PUT "http://ss.bscstorage.com/<Your-Bucket-Name>/?KID=baishan,<acce
```

DELETE Bucket

- Description: delete the specified Bucket.
- Note: Cannot delete the non-empty Bucket.
- Request Syntax:

```
DELETE /?formatter=json HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #Please refer to signature algorithm
```

or

```
DELETE /<Your-Bucket-Name>/?formatter=json HTTP/1.1
Host: ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #Please refer to signature algorithm
```

- Response (no HTTP body)：

```
HTTP/1.1 204 No Content
Date: Tue, 08 Apr 2014 02:59:47 GMT
Content-Length: 0
Connection: keep-alive
X-RequestId: 00078d50-1404-0810-5947-782bcb10b128
X-Requester: Your UserId
```

- Request example:

```
curl -v -X DELETE -H “Date: Sat, 20 Nov 2286 17:46:39 GMT” -H “Authorization:
```

or

```
curl -v -X DELETE "http://<Your-Bucket-Name>.ss.bscstorage.com/?KID=baishan,<a
```

or

```
curl -v -X DELETE "http://ss.bscstorage.com/<Your-Bucket-Name>/?KID=baishan,<a
```

PUT Bucket ACL

- Description: sets ACL rules for the specified Bucket. For more information, please refer to： 《ACL》
- Request Syntax:

```
PUT /?acl&formatter=json HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #Please refer to signature algorithm

#ACL规则
{
    'Baishan0000000000000001' :  [ "read", "read_acp" , "write", "write_acp" ]
    'GRPS000000ANONYMOUSE' :  [ "read", "read_acp" , "write", "write_acp" ],
    'GRPS0000000CANONICAL' :  [ "read", "read_acp" , "write", "write_acp" ],
}
```

- Response:

```
HTTP/1.1 200 OK
Date: Tue, 08 Apr 2014 02:59:47 GMT
Connection: keep-alive
X-RequestId: 00078d50-1404-0810-5947-782bcb10b128
X-Requester: Your UserId
{
    "Owner": {
        "DisplayName": "",
        "ID": "user_authed"
    },
    "AccessControlList": {
        "Grant": [
            {
                "Grantee": {
                    "DisplayName": "",
                     "ID": "Baishan0000000000000001"
                },
                "Permission": "READ,READ_ACP,WRITE,WRITE_ACP"
            },

            ...
        ]
    }
}
```

- Please refer to： 《ACL》
- Request example:

```
curl -v -T "acl.txt" -H "Date: Sat, 20 Nov 2286 17:46:39 GMT" -H "Authorizatio
```

GET Bucket ACL

- Description: get ACL information of the specified bucket. For more information, please refer to:《ACL》)
- Request format:

```
GET /?acl&formatter=json HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #Please refer to signature algorithm
```

or

```
GET /<Your-Bucket-Name>/?acl&formatter=json HTTP/1.1
Host: ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #Please refer to signature algorithm
```

Response:

```
HTTP/1.1 200 OK
Date: Tue, 08 Apr 2014 02:59:47 GMT
Content-Length: 123
Connection: keep-alive
X-RequestId: 00078d50-1404-0810-5947-782bcb10b128
X-Requester: Your UserId

{
    "Owner": {
        "DisplayName": "",
        "ID": "Baishan0000000000000001"
    },

    "AccessControlList": {
        "Grant": [
            {
                "Grantee": {
                    "DisplayName": "",
                    "ID": "GRPS000000ANONYMOUSE"
                },
                "Permission": "READ"
            },
            {
                "Grantee": {
                    "DisplayName": "",
                    "ID": "Baishan0000001001NHT3M7"
                },
                "Permission": "READ, READ_ACP, WRITE, WRITE_ACP"
            },
            {
                "Grantee": {
                    "DisplayName": "",
                    "ID": "Baishan0000000000000001"
                },
                "Permission": "READ, WRITE"
            },

            ...
        ]
    }
}
```

- Please refer to：《ACL》)
- Request example：

```
curl -v -H "Date: Sat, 20 Nov 2286 17:46:39 GMT" -H "Authorization: Baishan <a
```

or

```
curl -v "http://<Your-Bucket-Name>.ss.bscstorage.com/?acl&KID=baishan,<access_
```

or

```
curl -v "http://ss.bscstorage.com/<Your-Bucket-Name>/?acl&KID=baishan,<access_
```

Object Operations

HEAD Object

- Description: use the HEAD request method to obtain the Metadata of the Object.
- Request format:

```
HEAD /<ObjectName> HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #Please refer to signature algorithm
```

- Response（No HTTP body）：

```
HTTP/1.1 200 OK
Date: Tue, 08 Apr 2014 02:59:47 GMT
Connection: keep-alive
Content-Type: <object-mime-type>
Content-Length: <object-file-bytes>
ETag: "<MD5 value of file>"
Last-Modified: <Last modification time>
x-amz-request-id: 0000106c-1608-0810-4621-00163e000064
x-amz-s2-requester: <Your UserName>
x-amz-meta-foo1: <value1> #custom meta: foo1
x-amz-meta-foo2: <value2> #custom meta: foo2
```

- Request Headers：

Name	Description	Required
Range	Downloads the specified range bytes of an object. For more information about the HTTP Range header, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35. • Type: String • Default: None • Constraints: None	No
If-Modified-Since	Return the object only if it has been modified since the specified time, otherwise return a 304 (not modified). • Type: String • Default: None • Constraints: None	No
If-Unmodified-Since	Return the object only if it has not been modified since the specified time, otherwise return a 412 (precondition failed). • Type: String • Default: None • Constraints: None	No
If-Match	Return the object only if its entity tag (ETag) is the same as the one specified; otherwise, return a 412 (precondition failed). • Type: String • Default: None • Constraints: None	No
If-None-Match	Return the object only if its entity tag (ETag) is different from the one specified; otherwise, return a 304 (not modified). • Type: String • Default: None • Constraints: None	No

- Response Headers：

Name	Description
Content-Type	Object's mime-type
Content-Length	Object's Size(bytes)
ETag	Object's hash value，normally is md5 value
Last-Modified	Object last modified time
x-amz-meta-*	User can customize file property information, return value when read For example： x-amz-meta-UploadLocation: My Home X-amz-meta-ReviewedBy: test@test.net X-amz-meta-FileChecksum: 0x02661779 X-amz-meta-CheckSumAlgorithm: crc32
x-amz-meta-crc32	Object's CRC32 value

- Request example:

```
curl -v -X HEAD -H "Date: Sat, 20 Nov 2286 17:46:39 GMT" -H "Authorization: AW
```

GET Object

- Description: get an object (download).
- Request format:

```
GET /<ObjectName> HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #Please refer to signature algorithm
Range: bytes=<byte_range> #Support breakpoint Download
```

- Response:

```
HTTP/1.1 200 OK
Date: Tue, 08 Apr 2014 02:59:47 GMT
Connection: keep-alive
Content-Type: <object-mime-type>
Content-Length: <object-file-bytes>
ETag: "<MD5 value of file>"
Last-Modified: <Last modification time>
x-amz-request-id: 0000106c-1608-0810-4621-00163e000064
x-amz-s2-requester: <Your UserName>
x-amz-meta-foo1: <value1> #Custom meta: foo1
x-amz-meta-foo2: <value2> #Custom meta: foo2

#File content
<BODY>
```

- Request Headers：

Name	Description	Required
Range	Downloads the specified range bytes of an object. For more information about the HTTP Range header, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35. • Type: String • Default: None • Constraints: None	No
If-Modified-Since	Return the object only if it has been modified since the specified time, otherwise return a 304 (not modified). • Type: String • Default: None • Constraints: None	No
If-Unmodified-Since	Return the object only if it has not been modified since the specified time, otherwise return a 412 (precondition failed). • Type: String • Default: None • Constraints: None	No
If-Match	Return the object only if its entity tag (ETag) is the same as the one specified; otherwise, return a 412 (precondition failed). • Type: String • Default: None • Constraints: None	No
If-None-Match	Return the object only if its entity tag (ETag) is different from the one specified; otherwise, return a 304 (not modified). • Type: String • Default: None • Constraints: None	No

- Response Headers：

Name	Description
Content-Type	Object's mime-type
Content-Length	Object's size(bytes)
ETag	Object's hash value，normally it is md5
Last-Modified	Object's last modified time
x-amz-meta-*	You can customize the file attribute information and return the original value when reading. For example： x-amz-meta-UploadLocation: My Home X-amz-meta-ReviewedBy: test@test.net X-amz-meta-FileChecksum: 0x02661779 X-amz-meta-CheckSumAlgorithm: crc32
x-amz-meta-crc32	Object's CRC32 value

- Request example:

```
curl -v -H "Range: bytes=0-1024" -H "Date: Sat, 20 Nov 2286 17:46:39 GMT" -H "
```

- Application examples:
- Standard example:

```
GET /my_bucket/path/to/my/file.txt HTTP/1.1
Host: ss.bscstorage.com
Date: Sun, 1 Jan 2006 12:00:00 GMT
Authorization: AWS AccessKey:ssig
Range: bytes=100-2048
```

```
-    响应：
```

```
HTTP/1.1 206 Partial Content
Server: openresty/1.9.7.4
Content-Type: application/xml
Connection: keep-alive
x-amz-request-id: 0000106c-1608-0810-4621-00163e000064
x-amz-s2-requester: GRPS000000ANONYMOUSE
Date: Mon, 08 Aug 2016 02:46:21 GMT
Last-Modified: Mon, 08 Aug 2016 02:45:55 GMT
ETag: "21b1a992d1cbf49729fc4461e55dd94f"
x-amz-meta-s2-size: 109051904
x-amz-meta-s2-crc32: 9422bc32
Cache-Control: max-age=31536000
Content-Length: 109051904

...
file_content
...
```

- Download methods using various verification measures:

```
GET /path/to/my/file.txt?AWSAccessKeyId=<AccessKey>&Expires=<1175139620>&Signa
Host: my_bucket.ss.bscstorage.com
Date: date
Range: bytes=byte_range
```

- Response:

```
HTTP/1.1 206 Partial Content
Server: openresty/1.9.7.4
Content-Type: application/xml
Connection: keep-alive
x-amz-request-id: 0000106c-1608-0810-4621-00163e000064
x-amz-s2-requester: GRPS000000ANONYMOUSE
Date: Mon, 08 Aug 2016 02:46:21 GMT
Last-Modified: Mon, 08 Aug 2016 02:45:55 GMT
ETag: "21b1a992d1cbf49729fc4461e55dd94f"
x-amz-meta-s2-size: 109051904
x-amz-meta-s2-crc32: 9422bc32
Cache-Control: max-age=31536000
Content-Length: 109051904

...
file_content
...
```

For the meaning of QueryString in the above example, please refer to the description of authentication method in [signature algorithm] [1].

PUT Object

- Description: upload a file by post (based on browser form).
- Request：

```
POST / HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Content-Length: <length>

Content-Type:multipart/form-data; boundary=----WebKitFormBoundary1dIjDASRYXQm6

------WebKitFormBoundary1dIjDASRYXQm6DNA
Content-Disposition: form-data; name="key"

destinationProject/${filename}
------WebKitFormBoundary1dIjDASRYXQm6DNA
Content-Disposition: form-data; name="success_action_redirect"

http://123.abc.com/1.php?f=1111.txt
------WebKitFormBoundary1dIjDASRYXQm6DNA
Content-Disposition: form-data; name="AWSAccessKeyId"

00M414ZO0X30
------WebKitFormBoundary1dIjDASRYXQm6DNA
Content-Disposition: form-data; name="Policy"

eyJleHBpcmF0aW9uIjoiMjAxMi0wNi0wNlQwNjozOTo0MS4wMDBaIiwiY29uZGl0aW9ucyI6W3siYn
------WebKitFormBoundary1dIjDASRYXQm6DNA
Content-Disposition: form-data; name="Signature"

VK6Kw4kRqW2e84ZIX2cV2QqHo58=
------WebKitFormBoundary1dIjDASRYXQm6DNA
Content-Disposition: form-data; name="file"; filename="112233.txt"
Content-Type: text/plain

------WebKitFormBoundary1dIjDASRYXQm6DNA
Content-Disposition: form-data; name="submit"

上传
------WebKitFormBoundary1dIjDASRYXQm6DNA-
```

- Form elements:

Name	Description	Required
AWSAccessKeyId	It is the AccessKey, which can be obtained from the console	Yes
key	The key (path) after object upload, such as: angel / ${filename}, variable ${filename} will be automatically replaced with the file name of the uploaded file; of course, you can also directly specify the file name of the uploaded file stored in the storage, such as: angel / path / to / myfile.txt,Variable names can be: filename, SHA1, MD5, size	Yes
acl	ACL of file: set an ACL when creating a file. Please refer to 《ACL》	No
success_action_status	The response code after successful upload can be set to 200, 201, or 204 (default). If it is set to 200 or 204, the returned body is empty and the status is 200 or 204. If it is set to 201, the returned body in XML format and the status is 201. If it is set to an illegal value, the value is ignored and the default value 204 is used Note: if success is set_ action_ Redirect or redirect, this setting is ignored	No
success_action_redirect, redirect	The URL redirected by the client after successful upload. The actual returned location will add bucket, key and Etag querystring to the original URL	No
Policy	File strategy, JSON format string, and encoded with Base64. It will be described in detail later	Yes
Signature	The string signed with the secret key. It will be described in detail later	Yes
file	Input form with type = file	Yes
x-amz-meta-*	User defined metadeta. The header starts with x-amz-meta -, and all meta are stored in the form of key: value. The maximum limit is 64KB. It is returned as it is when HEAD or GET	No

Name	Description	Required
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	like put_file, refer to put_file interface	No

Construction of policy：

The policy is a JSON document encoded with UTF-8 and Base64, which specifies the conditions that the request must meet and is used to authenticate the content. Depending on how you design policy documents, you can use them for each upload, each user, all uploads, or other designs that meet your needs.

```
{
    "expiration": "2014-04-10T08:55:34.000Z",

    "conditions": [

        {
            "bucket": "my-bucket-name"
        },

        {
            "acl": "private"
        },

        [
            "starts-with", "$key", "my_prefix/"
        ],

        [
            "content-length-range", 0, 52428800
        ]
    ]
}
```

- Description of the above example:
    - Upload must be before "2014-04-10t08:55:34.000z".
    - Upload the file to the bucket named "my bucket name".
    - starts-with：$key must start with "my_ Prefix" / '(the "$key" in the policy must be preceded by "$"). If the $key value is empty, there is no prefix before the file name.
    - content-length-range：The file size must be within the specified range.
    - Finally, Base64 encode the policy and set it to the value of the form policy.expiration:

Expiration

Expiration is used to specify the expiration time of the policy. The expiration date of the policy is specified in ISO 8601 UTC date format. For example, "2007-12-01t12:00:00.000z" specifies that expiration is required in the policy.

Condition:

Condition is used to verify the content of the uploaded object and the fields filled in by the form

Condition type

Condition	Description
Precise Matching	1. Precise matching will verify that the field matches a specific value. This example instructs that the ACL must be set to public-read: {"acl": "public-read" } 2.ACL must be set to public-read alternative: ["eq", "$acl", "public-read"]
Starts With	If the value must start with a specific value, use starts-with. this example instructs that the key must start with user/betty starts with:["starts-with", "$key", "user/betty/"]
Specify the range	For fields that accept a range, use a comma to separate the upper and lower limit values. This example allows 1 to 10 MB of file sizes:["content-length-range", 1048579, 10485760], byte

Conditions field

The conditions in the policy document verify the content of the uploaded object. Each form field you specify in the form（AWSAccessKeyId、Signature、 file、Policy except）Can be used as a condition

Signature Construction:

- Encoding policy with UTF-8
- Encode the policy in UTF-8 form with Base64
- Use HMAC SHA-1 and your secret key to convert your policy. Finally, Base64 coding is performed. For example, when using PHP，base64_encode(hash_hmac("sha1", $policy, $SECRETKEY, true));
- Set the final value to the value of the form signature.

Finally generated HTML form:

```
<form method="post" action="http://my-bucket.<?=c('api_host');?>/" enctype="mu
    <input type="hidden" name="AWSAccessKeyId" value="您的accesskey" />
    <input type="hidden" name="key" value="my_prefix/${filename}" />
    <input type="hidden" name="acl" value="private" />
    <input type="hidden" name="success_action_status" value="201" />
    <input type="hidden" name="Policy" value="eyJleHBpcmF0aW9uIjoiMjAxNC0wNC0xl
    <input type="hidden" name="Signature" value="HnOSk3kfx5LFtn4CIiFcSglQUXc="
    <input type="file" name="file" />
    <input type="submit" value="上传" />
</form>
```

- matters needing attention：
  - The URI after the POST request can only be “/”

- success_action_redirect：Specify the URL to be redirected by the client after successful upload.
- key：The variable ${filename} will be automatically replaced with the file name of the uploaded file

APPEND Object

- Append Object Files can be uploaded by adding write. The added files can be files uploaded in any way in the bucket (except fragment files uploaded in fragments), such as put object, copy object, and files uploaded in fragments and merged.

  The append upload and put object interfaces are consistent, except for the following two points：

  - x-amz-meta-s2-append-position Request header, required, used to identify the location of append upload and upload files appended to the current file
  - When the value of position is 0, if the currently appended file does not exist or the size of the file is equal to 0, the data will be appended to the end of the file and a success will be returned, otherwise a 409 error will be returned;
  - When the value of position is greater than 0, if it is equal to the size of the current file, it returns success after appending the data to the starting position of size, and the modification time of the modified file is the current time; Otherwise, a 409 error is returned, and the size of the current file is set in the x-amz-meta-append-position header of the response.
  - X-amz-meta-s2-directive request header, values: COPY, REPLACE ptional (COPY), used to identify whether the append file is the file meta information of the file to be overwritten

    Attention：

  - Only one request of the same position can be appended successfully, and due to concurrency, setting the correct position may also return failure

  - A maximum of 2000 requests can be appended to the same file
  - The maximum file size for a single request is 20G
  - The total size of the appended file is 1T at most

- Request format:

```
PUT /<ObjectName> HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Content-Length: <object data length>
Content-Type: <mime-type>
x-amz-meta-s2-append-position: <position>
Authorization: <authorization string> #Please refer to signature algorithm
[object data]
```

Response:

```
HTTP/1.1 200 OK
Date: Tue, 08 Apr 2014 02:59:47 GMT
Connection: keep-alive
ETag: "<MD5 value of file>"
x-amz-request-id: 0000106c-1608-0810-4621-00163e000064
x-amz-s2-requester: <Your UserName>
x-amz-meta-s2-append-position: <position>
```

- Request Headers：

Name	Description	Rec
x-amz-meta-append-position	It is used to identify the location where append uploads and uploads are appended to the current file	Yes
x-amz-meta-s2-directive	Specifies whether to use the file meta of the attached file. The value is：COPY,REPALCE，The default is:COPY, If set to COPY，The file meta of the append file is used. If it is set to REPLACE, The file meta carried in the current append request is used	No
Expires	When the file expires, the system will automatically clear the file (not immediately, and the clearing time is irregular). Format reference：http://www.w3.org/Protocols/rfc2616/ rfc2616-sec14.html#sec14.21.	No
Cache-Control	file Cache，Standard HTTP protocol. For more information, see:http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.9	No
Content-Type	File MIME type. The original value is returned when reading	No
Content-Length	File size, original value returned when reading	Yes
Content-MD5	Base64 encoded file MD5 (fail when inconsistent with the transmitted content). Note: the string format is the value encoded by Base64 in RFC standard	No
Content-Disposition	HTTP standard file attribute information. The original value is returned when reading. See:http://www.w3.org/Protocols/rfc2616/rfc2616-sec19.html#sec19.5.1	No
Content-Encoding	File code, HTTP standard file attribute information, and the original value is returned when reading. See:http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.11	No
x-amz-acl	File ACL: set an ACL while creating a file. Please refer to 《ACL》	No
x-amz-meta-*	User defined metadeta. The header starts with x-amz-meta -, and all meta are stored in the form of key: value. The maximum limit is 64KB. It is returned as it is when HEAD or GET	No

POST Object

- Description: upload a file by post (based on browser form).
- Request：

```
POST / HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Content-Length: <length>

Content-Type:multipart/form-data; boundary=----WebKitFormBoundary1dIjDASRYXQm6

------WebKitFormBoundary1dIjDASRYXQm6DNA
Content-Disposition: form-data; name="key"

destinationProject/${filename}
------WebKitFormBoundary1dIjDASRYXQm6DNA
Content-Disposition: form-data; name="success_action_redirect"

http://123.abc.com/1.php?f=1111.txt
------WebKitFormBoundary1dIjDASRYXQm6DNA
Content-Disposition: form-data; name="AWSAccessKeyId"

00M414ZO0X30
------WebKitFormBoundary1dIjDASRYXQm6DNA
Content-Disposition: form-data; name="Policy"

eyJleHBpcmF0aW9uIjoiMjAxMi0wNi0wNlQwNjozOTo0MS4wMDBaIiwiY29uZGl0aW9ucyI6W3siYn
------WebKitFormBoundary1dIjDASRYXQm6DNA
Content-Disposition: form-data; name="Signature"

VK6Kw4kRqW2e84ZIX2cV2QqHo58=
------WebKitFormBoundary1dIjDASRYXQm6DNA
Content-Disposition: form-data; name="file"; filename="112233.txt"
Content-Type: text/plain

------WebKitFormBoundary1dIjDASRYXQm6DNA
Content-Disposition: form-data; name="submit"

上传
------WebKitFormBoundary1dIjDASRYXQm6DNA-
```

- Form elements:

Name	Description	Required
AWSAccessKeyId	It is the AccessKey, which can be obtained from the console	Yes
key	The key (path) after object upload, such as: angel / ${filename}, variable ${filename} will be automatically replaced with the file name of the uploaded file; of course, you can also directly specify the file name of the uploaded file stored in the storage, such as: angel / path / to / myfile.txt,Variable names can be: filename, SHA1, MD5, size	Yes
acl	ACL of file: set an ACL when creating a file. Please refer to 《ACL》	No
success_action_status	The response code after successful upload can be set to 200, 201, or 204 (default). If it is set to 200 or 204, the returned body is empty and the status is 200 or 204. If it is set to 201, the returned body in XML format and the status is 201. If it is set to an illegal value, the value is ignored and the default value 204 is used Note: if success is set_ action_ Redirect or redirect, this setting is ignored	No
success_action_redirect, redirect	The URL redirected by the client after successful upload. The actual returned location will add bucket, key and Etag querystring to the original URL	No
Policy	File strategy, JSON format string, and encoded with Base64. It will be described in detail later	Yes
Signature	The string signed with the secret key. It will be described in detail later	Yes
file	Input form with type = file	Yes
x-amz-meta-*	User defined metadeta. The header starts with x-amz-meta -, and all meta are stored in the form of key: value. The maximum limit is 64KB. It is returned as it is when HEAD or GET	No

Name	Description	Required
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	like put_file, refer to put_file interface	No

Construction of policy：

The policy is a JSON document encoded with UTF-8 and Base64, which specifies the conditions that the request must meet and is used to authenticate the content. Depending on how you design policy documents, you can use them for each upload, each user, all uploads, or other designs that meet your needs.

```
{
    "expiration": "2014-04-10T08:55:34.000Z",

    "conditions": [

        {
            "bucket": "my-bucket-name"
        },

        {
            "acl": "private"
        },

        [
            "starts-with", "$key", "my_prefix/"
        ],

        [
            "content-length-range", 0, 52428800
        ]
    ]
}
```

- Description of the above example:
    - Upload must be before "2014-04-10t08:55:34.000z".
    - Upload the file to the bucket named "my bucket name".
    - starts-with：$key must start with "my_ Prefix" / '(the "$key" in the policy must be preceded by "$"). If the $key value is empty, there is no prefix before the file name.
    - content-length-range：The file size must be within the specified range.
    - Finally, Base64 encode the policy and set it to the value of the form policy.expiration:

Expiration

Expiration is used to specify the expiration time of the policy. The expiration date of the policy is specified in ISO 8601 UTC date format. For example, "2007-12-01t12:00:00.000z" specifies that expiration is required in the policy.

Condition:

Condition is used to verify the content of the uploaded object and the fields filled in by the form

Condition type

Condition	Description
Precise Matching	1. Precise matching will verify that the field matches a specific value. This example instructs that the ACL must be set to public-read: {"acl": "public-read" } 2.ACL must be set to public-read alternative: ["eq", "$acl", "public-read"]
Starts With	If the value must start with a specific value, use starts-with. this example instructs that the key must start with user/betty starts with:["starts-with", "$key", "user/betty/"]
Specify the range	For fields that accept a range, use a comma to separate the upper and lower limit values. This example allows 1 to 10 MB of file sizes:["content-length-range", 1048579, 10485760], byte

Conditions field

The conditions in the policy document verify the content of the uploaded object. Each form field you specify in the form（AWSAccessKeyId、Signature、 file、Policy except）Can be used as a condition

Signature Construction:

- Encoding policy with UTF-8
- Encode the policy in UTF-8 form with Base64
- Use HMAC SHA-1 and your secret key to convert your policy. Finally, Base64 coding is performed. For example, when using PHP，base64_encode(hash_hmac("sha1", $policy, $SECRETKEY, true));
- Set the final value to the value of the form signature.

Finally generated HTML form:

```
<form method="post" action="http://my-bucket.<?=c('api_host');?>/" enctype="mu
    <input type="hidden" name="AWSAccessKeyId" value="您的accesskey" />
    <input type="hidden" name="key" value="my_prefix/${filename}" />
    <input type="hidden" name="acl" value="private" />
    <input type="hidden" name="success_action_status" value="201" />
    <input type="hidden" name="Policy" value="eyJleHBpcmF0aW9uIjoiMjAxNC0wNC0xl
    <input type="hidden" name="Signature" value="HnOSk3kfx5LFtn4CIiFcSglQUXc="
    <input type="file" name="file" />
    <input type="submit" value="上传" />
</form>
```

- matters needing attention：
  - The URI after the POST request can only be “/”

- success_action_redirect：Specify the URL to be redirected by the client after successful upload.
- key：The variable ${filename} will be automatically replaced with the file name of the uploaded file

PUT Object - Copy

- Description: create an object by copying (copy another file in the system instead of uploading specific file content).
- Request format:：

```
PUT /<ObjectName> HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
x-amz-copy-source: </source-bucket/source-object>
Authorization: <authorization string> #Please refer to signature algorithm
```

- response：

```
HTTP/1.1 200 OK
Date: Tue, 08 Apr 2014 02:59:47 GMT
Connection: keep-alive
ETag: "<MD5 value of file>"
x-amz-request-id: 0000106c-1608-0810-4621-00163e000064
x-amz-s2-requester: <Your UserName>

<?xml version="1.0" encoding="UTF-8"?>
<CopyObjectResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <LastModified>Mon, 08 Aug 2016 05:04:10 GMT</LastModified>
    <ETag>870c06c00566c4fb1861bb10f34d1904</ETag>
</CopyObjectResult>
```

- Request Headers：

Name	Description	Re
x-amz-copy-source	The address of the file to be copied. Format: / source bucket / source object. urlencode the whole.	Yes
Cache-Control	File cache, standard HTTP protocol. For more information, see:http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.9	No
Expires	The cache expiration time of the file in the client or browser allows the client not to check the server before this time, and the original value is returned when reading. Format is：Sun, 29 Jul 2018 20:36:14 UTC	No
Content-Type	File mime type. The original value is returned when reading	No
Content-Length	must be 0	=0
Content-Disposition	HTTP standard file attribute information. The original value is returned when reading. See：http://www.w3.org/Protocols/rfc2616/rfc2616-sec19.html#sec19.5.1	No
Content-Encoding	File code, HTTP standard file attribute information, and the original value is returned when reading. See:http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.11	No
x-amz-acl	File ACL: set an ACL while creating a file. Please refer to 《ACL》	No
x-amz-meta-*	User defined metadeta. The header starts with x-amz-meta -, and all meta are stored in the form of key: value. The maximum limit is 64KB. It is returned as it is when HEAD or GET	No
x-amz-copy-source-if-match	If the specified Etag matches the Etag of the source file, the source file can be copied, otherwise 412 is returned(PreconditionFailed)	No
x-amz-copy-source-if-nonematch	If the specified Etag does not match the Etag of the source file, the source file can be copied, otherwise 412 is returned(PreconditionFailed)	No
x-amz-copy-source-if-unmodified-since	If the source file has not been modified since the specified time, you can copy the source file; otherwise, return412(PreconditionFailed)	No
x-amz-copy-source-if-modified-since	If the source file has been modified since the specified time, you can copy the source file, otherwise return412(PreconditionFailed)	No

- Response Body（Response XML Body）：

Name	Description
CopyObjectResult	Contains Etag and LastModified elements
ETag	File's ETag
CopyObjectResult	Last modification time of the file

- Request example: ``` curl -v -X PUT -H "x-amz-copy-source: /bucket-123/path/to/file123.txt" -H "Date:

Sat, 20 Nov 2286 17:46:39 GMT" -H "Authorization: AWS :" "http://.ss.bscstorage.com/path/to/myfile.txt" ```

DELETE Object

- Description: deletes the specified object.
- Request format:

```
DELETE /<ObjectName> HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #Please refer to signature algorithm
```

- Response (no HTTP Body)

```
HTTP/1.1 204 No Content
Date: Tue, 08 Apr 2014 02:59:47 GMT
Content-Length: 0
Connection: keep-alive
x-amz-request-id: 0000106c-1608-0810-4621-00163e000064
x-amz-s2-requester: <Your UserName>
```

- Request example:

```
curl -v -X DELETE -H "Date: Sat, 20 Nov 2286 17:46:39 GMT" -H "Authorization:
```

GET Object

- Description: get an object (download).
- Request format:

```
GET /<ObjectName> HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #Please refer to signature algorithm
Range: bytes=<byte_range> #Support breakpoint Download
```

- Response:

```
HTTP/1.1 200 OK
Date: Tue, 08 Apr 2014 02:59:47 GMT
Connection: keep-alive
Content-Type: <object-mime-type>
Content-Length: <object-file-bytes>
ETag: "<MD5 value of file>"
Last-Modified: <Last modification time>
x-amz-request-id: 0000106c-1608-0810-4621-00163e000064
x-amz-s2-requester: <Your UserName>
x-amz-meta-foo1: <value1> #Custom meta: foo1
x-amz-meta-foo2: <value2> #Custom meta: foo2

#File content
<BODY>
```

- Request Headers：

Name	Description	Required
Range	Downloads the specified range bytes of an object. For more information about the HTTP Range header, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35. • Type: String • Default: None • Constraints: None	No
If-Modified-Since	Return the object only if it has been modified since the specified time, otherwise return a 304 (not modified). • Type: String • Default: None • Constraints: None	No
If-Unmodified-Since	Return the object only if it has not been modified since the specified time, otherwise return a 412 (precondition failed). • Type: String • Default: None • Constraints: None	No
If-Match	Return the object only if its entity tag (ETag) is the same as the one specified; otherwise, return a 412 (precondition failed). • Type: String • Default: None • Constraints: None	No
If-None-Match	Return the object only if its entity tag (ETag) is different from the one specified; otherwise, return a 304 (not modified). • Type: String • Default: None • Constraints: None	No

- Response Headers：

Name	Description
Content-Type	Object's mime-type
Content-Length	Object's size(bytes)
ETag	Object's hash value，normally it is md5
Last-Modified	Object's last modified time
x-amz-meta-*	You can customize the file attribute information and return the original value when reading. For example： x-amz-meta-UploadLocation: My Home X-amz-meta-ReviewedBy: test@test.net X-amz-meta-FileChecksum: 0x02661779 X-amz-meta-CheckSumAlgorithm: crc32
x-amz-meta-crc32	Object's CRC32 value

- Request example:

```
curl -v -H "Range: bytes=0-1024" -H "Date: Sat, 20 Nov 2286 17:46:39 GMT" -H "
```

- Application examples:
- Standard example:

```
GET /my_bucket/path/to/my/file.txt HTTP/1.1
Host: ss.bscstorage.com
Date: Sun, 1 Jan 2006 12:00:00 GMT
Authorization: AWS AccessKey:ssig
Range: bytes=100-2048
```

```
-    响应：
```

```
HTTP/1.1 206 Partial Content
Server: openresty/1.9.7.4
Content-Type: application/xml
Connection: keep-alive
x-amz-request-id: 0000106c-1608-0810-4621-00163e000064
x-amz-s2-requester: GRPS000000ANONYMOUSE
Date: Mon, 08 Aug 2016 02:46:21 GMT
Last-Modified: Mon, 08 Aug 2016 02:45:55 GMT
ETag: "21b1a992d1cbf49729fc4461e55dd94f"
x-amz-meta-s2-size: 109051904
x-amz-meta-s2-crc32: 9422bc32
Cache-Control: max-age=31536000
Content-Length: 109051904

...
file_content
...
```

- Download methods using various verification measures:

```
GET /path/to/my/file.txt?AWSAccessKeyId=<AccessKey>&Expires=<1175139620>&Signa
Host: my_bucket.ss.bscstorage.com
Date: date
Range: bytes=byte_range
```

- Response:

```
HTTP/1.1 206 Partial Content
Server: openresty/1.9.7.4
Content-Type: application/xml
Connection: keep-alive
x-amz-request-id: 0000106c-1608-0810-4621-00163e000064
x-amz-s2-requester: GRPS000000ANONYMOUSE
Date: Mon, 08 Aug 2016 02:46:21 GMT
Last-Modified: Mon, 08 Aug 2016 02:45:55 GMT
ETag: "21b1a992d1cbf49729fc4461e55dd94f"
x-amz-meta-s2-size: 109051904
x-amz-meta-s2-crc32: 9422bc32
Cache-Control: max-age=31536000
Content-Length: 109051904

...
file_content
...
```

For the meaning of QueryString in the above example, please refer to the description of authentication method in [signature algorithm] [1].

PUT Object ACL

- Description: sets ACL rules for the specified Object. For more information, please refer to： 《ACL》
- Request format:

```
PUT /<ObjectName>?acl HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #Please refer to signature algorithm

#ACL规则: XML or Json
{
    "Baishan0000000000000001" :  [ "read", "read_acp" , "write", "write_acp" ]
    "GRPS000000ANONYMOUSE" :  [ "read", "read_acp" , "write", "write_acp" ],
    "GRPS0000000CANONICAL" :  [ "read", "read_acp" , "write", "write_acp" ]
}
```

- Request Headers：

Name	Description	Required
x-amz-acl	File ACL: please refer to 《ACL》	No

```
HTTP/1.1 200 OK
Date: Tue, 08 Apr 2014 02:59:47 GMT
Connection: keep-alive
x-amz-request-id: 0000106c-1608-0810-4621-00163e000064
x-amz-s2-requester: <Your UserName>
```

- Request Body（请求Body）： XML or JSon format acl body

Note: If x-amz-acl is specified, the header will ignore the ACL in the body. The XML format in the body is determined by the request parameter formatter parameter, and the default is XML

- Response（no HTTP Body）：
- Please refer to： 《ACL》
- Request example：

```
curl -v -T "acl.txt" -H "Date: Sat, 20 Nov 2286 17:46:39 GMT" -H "Authorizatio
AWS <access_key>:<ssig>" "http://<Your-Bucket-Name>.ss.bscstorage.com/path/to/
```

Initiate Multipart Upload

- Description: initialization interface for large file fragment upload
- Note: user authentication is required in initializing the upload interface. Anonymous users cannot use this interface. During the initial upload, the meta binding information required for file upload can be given. This information will be retained in subsequent uploads and written to the cloud storage system when the final upload is completed.
- Request format:

```
POST /<ObjectName>?uploads HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Content-Type: <mime-type>
x-amz-meta-foo1: <value1> #custom meta: foo1
x-amz-meta-foo2: <value2> #custom meta: foo2
Authorization: <authorization string> #Please refer to signature algorithm
```

- Response:

```
HTTP/1.1 200 OK
Date: Tue, 08 Apr 2014 02:59:47 GMT
Connection: keep-alive
x-amz-request-id: 0000106c-1608-0810-4621-00163e000064
x-amz-s2-requester: <Your UserName>

<?xml version="1.0" encoding="UTF-8"?>
<InitiateMultipartUploadResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/"
    <Bucket>your-bucket</Bucket>
    <Key>objectName</Key>
    <UploadId>VXBsb2FkIElEIGZvciA2aWWpbmcncyBteS1tb3ZpZS5tMnRzIHVwbG9hZA</Uplo
</InitiateMultipartUploadResult>
```

- Request Headers： reference resources for put_object documentation
- Response Body（Response XML Body）：

Name	Description
InitiateMultipartUploadResult	Contains Bucket, Key and UploadId elements
Bucket	bucket name
Key	objectName
UploadId	Identify the ID of the block upload. This parameter needs to be carried when uploading the block later

- Request example:

```
curl -v -X POST "Date: Sat, 20 Nov 2286 17:46:39 GMT" -H "Authorization: AWS <
```

Upload Part

- Description: upload slice interface
- Request format:

```
PUT /<ObjectName>?partNumber=<PartNumber>&uploadId=<UploadId> HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Content-Length: <Content-Length>
Content-MD5: <Content-MD5>
Authorization: <authorization string> #Please refer to signature algorithm

...
file_content
...
```

- Response:

```
HTTP/1.1 200 OK
Date: Tue, 08 Apr 2014 02:59:47 GMT
Connection: keep-alive
x-amz-request-id: 0000106c-1608-0810-4621-00163e000064
x-amz-s2-requester: <Your UserName>
Etag: <Etag>
```

- Request Parameters：

Parameter	Description	Required
partNumber	The serial number of the document slice, starting from 1	Yes
uploadId	Uploadid value obtained through Initiate Multipart Upload (large file fragment upload initialization interface)	Yes

- Request Heades：

Name	Description	Required
Content-Length	File size, original value returned when reading	Yes
Content-MD5	Base64 encoded file MD5 (fail when inconsistent with the transmitted content). Note: the string format is the value encoded by Base64 in RFC standard	No

- Attention：
  - The number of slices cannot exceed 2000.

Upload Part - Copy

- Description: create and upload a fragment by copying (do not upload specific file content, but copy another file in the system by COPY).
- Request format:

```
PUT /<ObjectName>?partNumber=PartNumber&uploadId=UploadId HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
x-amz-copy-source: </source-bucket/source-object>
Authorization: <authorization string> #Please refer to signature algorithm
```

- Response:

```
HTTP/1.1 200 OK
Date: Tue, 08 Apr 2014 02:59:47 GMT
Connection: keep-alive
ETag: "<文件的MD5值>"
x-amz-request-id: 0000106c-1608-0810-4621-00163e000064
x-amz-s2-requester: <Your UserName>

<?xml version="1.0" encoding="UTF-8"?>
<CopyObjectResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <LastModified>Mon, 08 Aug 2016 05:04:10 GMT</LastModified>
    <ETag>870c06c00566c4fb1861bb10f34d1904</ETag>
</CopyObjectResult>
```

- Request Parameters：

Parameter	Description	Required
partNumber	The serial number of the document slice, starting from 1	Yes
uploadId	Uploadid value obtained through Initiate Multipart Upload (large file fragment upload initialization interface)	Yes

- Request Headers：

Name	Description	Required
x-amz-copy-source	The address of the file to be copied. format：/source-bucket/source-object，We need to encode urlencode as a whole	Yes
Content-Length	Must be 0	=0
x-amz-copy-source-if-match	If the specified Etag matches the Etag of the source file, the source file can be copied, otherwise it returns412(PreconditionFailed)	No
x-amz-copy-source-if-nonematch	If the specified Etag does not match the Etag of the source file, the source file can be copied, otherwise it returns412(PreconditionFailed)	No
x-amz-copy-source-if-unmodified-since	If the source file has not been modified since the specified time, you can copy the source file; otherwise, return412(PreconditionFailed)	No
x-amz-copy-source-if-modified-since	If the source file is modified at the specified time, the source file can be copied, otherwise 412 is returned(PreconditionFailed)	No

- Response Body（Response XML Body）：

Name	Description
CopyObjectResult	contain ETag and LastModified
ETag	File's ETag
CopyObjectResult	File last modified time

- Request example:

```
curl -v -X PUT -H "x-amz-copy-source: /bucket-123/path/to/file123.txt" -H "Dat
Sat, 20 Nov 2286 17:46:39 GMT" -H "Authorization: AWS <access_key>:<ssig>" "ht
```

Complete Multipart Upload

- Description: large file fragment upload and splicing completion interface (merging interface)
- Request format:

```
POST /<ObjectName>?uploadId=<UploadId> HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Content-Type: text/json
Authorization: <authorization string> #Please refer to signature algorithm

<CompleteMultipartUpload>
    <Part>
        <PartNumber>1</PartNumber>
        <ETag>"a54357aff0632cce46d942af68356b38"</ETag>
    </Part>
    <Part>
        <PartNumber>2</PartNumber>
        <ETag>"0c78aef83f66abc1fa1e8477f296d394"</ETag>
    </Part>
    <Part>
        <PartNumber>3</PartNumber>
        <ETag>"acbd18db4cc2f85cedef654fccc4a4d8"</ETag>
    </Part>
</CompleteMultipartUpload>
```

- Response:

```
HTTP/1.1 200 OK
Date: Tue, 08 Apr 2014 02:59:47 GMT
Connection: keep-alive
x-amz-request-id: 0000106c-1608-0810-4621-00163e000064
x-amz-s2-requester: <Your UserName>

<?xml version="1.0" encoding="UTF-8"?>
<CompleteMultipartUploadResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/"
    <Location>http://Example-Bucket.ss.bscstorage.com/Example-Object</Location
    <Bucket>Example-Bucket</Bucket>
    <Key>Example-Object</Key>
    <ETag>"3858f62230ac3c915f300c664312c11f-9"</ETag>
</CompleteMultipartUploadResult>
```

- Request Parameters：

Name	Description	Required
PartNumber	Uploadid value obtained through Initiate Multipart Upload (large file fragment upload initialization interface)	Yes
ETag	The Etag value in the response header returned after successful upload through the Upload Part	Yes

  - Response Body（Response XML Body）：

Name	Description	Required
CompleteMultipartUploadResult	Contain Location, Bucket, Key and ETag element	Yes
Location	URI Identifies the newly uploaded file	Yes
Bucket	bucket name	Yes
Key	objectName	Yes
ETag	Etag of the file, the Etag and put_ Unlike the Etag of object, it is not the MD5 value of the file	Yes

List Parts

- Description: lists all blocks that have been uploaded
- Request format:

```
GET /<ObjectName>?uploadId=<UploadId> HTTP/1.1
Host: <Your-Bucket-Name>.ss.bscstorage.com
Date: <date>
Authorization: <authorization string> #Please refer to signature algorithm
```

- Response:

```
HTTP/1.1 200 OK
Date: Tue, 08 Apr 2014 02:59:47 GMT
Connection: keep-alive
x-amz-request-id: 0000106c-1608-0810-4621-00163e000064
x-amz-s2-requester: <Your UserName>

<?xml version="1.0" encoding="UTF-8"?>
<ListPartsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Bucket>example-bucket</Bucket>
<Key>example-object</Key>
<UploadId>XXBsb2FkIElEIGZvciBlbHZpbmcncyVcdS1tb3ZpZS5tMnRzEEEwbG9hZA</UploadId
<Initiator>
    <ID>initiator</ID>
    <DisplayName></DisplayName>
</Initiator>
<Owner>
    <ID>owner</ID>
    <DisplayName></DisplayName>
</Owner>
<StorageClass>STANDARD</StorageClass>
<PartNumberMarker>1</PartNumberMarker>
<NextPartNumberMarker>3</NextPartNumberMarker>
<MaxParts>2</MaxParts>
<IsTruncated>true</IsTruncated>
<Part>
    <PartNumber>2</PartNumber>
    <LastModified>2010-11-10T20:48:34.000Z</LastModified>
    <ETag>"7778aef83f66abc1fa1e8477f296d394"</ETag>
    <Size>10485760</Size>
</Part>
<Part>
    <PartNumber>3</PartNumber>

    <LastModified>2010-11-10T20:48:33.000Z</LastModified>
    <ETag>"aaaa18db4cc2f85cedef654fccc4a4x8"</ETag>
    <Size>10485760</Size>
</Part>
</ListPartsResult>
```

- Request Parameters：

Parameter	Description	Required
uploadId	Uploadid value obtained through Initiate Multipart Upload (large file fragment upload initialization interface)	Yes
max-parts	Returns the maximum number of parts, 2048 by default	Yes
part-number-marker	List the parts after the partnumber, excluding the part specified by the partnumber, which starts from 1	Yes

- Request Headers：
- Response Body（Response XML Body）：

Name	Description
ListPartsResult	Contain Bucket, Key, UploadId, Initiator, Owner, StorageClass, PartNumberMarker, NextPartNumberMarker, MaxParts, IsTruncated, Part element
Bucket	bucket name
Key	objectName
UploadId	Identify the ID of the block upload. This parameter needs to be carried when uploading the block later
Initiator	Include ID, DisplayName element
ID	init UserName
DisplayName	
Owner	Include ID, DisplayName element, specific file's owner
StorageClass	storage class
PartNumberMarker	list initial partNumber, does not include this partNumber
NextPartNumberMarker	Start the partNumber of the list next time
MaxParts	Maximum part numbers
IsTruncated	Is it truncated
Part	It contains partnumber, LastModified, Etag and size elements to represent the information of a part
PartNumber	Number of part
LastModified	Last modification time of part
ETag	part's md5
Size	part's size

Provisional signature：STS

Temporary voucher usage process

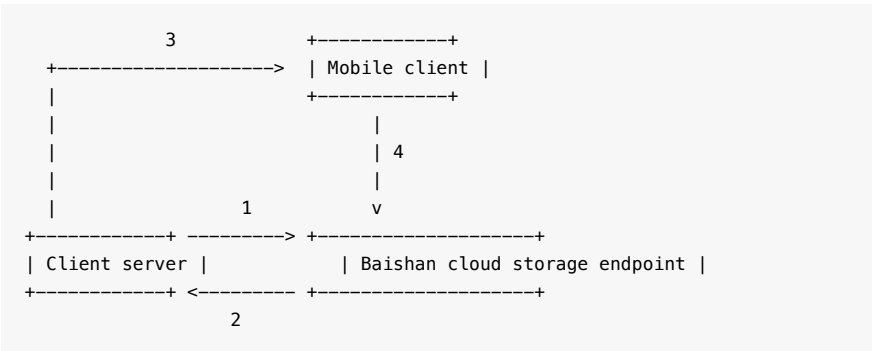
After opening the storage account in Baishan cloud storage, you can get the access key and secret key required to access the storage service. This is a permanent access certificate and has any operation permission. Therefore, in order to prevent the disclosure of the permanent certificate, it is not suitable to store it in unsafe places such as mobile clients. When you need to access the storage service from the mobile client, you can use the temporary certificate. At this time, the permanent certificate is stored on the client server. The client server uses the permanent certificate to request the temporary certificate from the Baishan cloud storage node, and then sends the obtained temporary certificate to the mobile client. The mobile client then uses the temporary certificate to access the storage service.

Temporary vouchers include three parts: temporary `access_key`, temporary `secret_key`, token. The method of using temporary credentials is similar to that of using permanent credentials, except that an additional token needs to be added. If SDK is used, this token can be passed in through parameters.

Temporary vouchers have a validity period. You can set the length of the validity period when obtaining temporary vouchers, which can be set to 15 minutes to 60 minutes.

The permissions of temporary vouchers can also be set when obtaining them. For example, they can be set to upload only, download only, or upload only to the specified directory.

STS使用流程



1. The client server accesses the STS service provided by the Baishan cloud storage node to obtain temporary credentials.
2. The cloud storage STS service returns temporary credentials.
3. The client server sends temporary credentials to the mobile client.
4. Mobile clients use temporary credentials to access cloud storage services.

Example

http://sts.ss.bscstorage.com

- Use the SDK of STS Service to access Baishan cloud storage STS Service. The endpoint is `http://sts.ss.bscstorage.com`

The following three parameters are commonly used to apply for temporary vouchers:

- `Policy` : used to describe the allocated permissions. It is a JSON string, which only `Statement` needs to be filled in field
  - `Effect` : `Allow` indicates allow permission and `Deny` indicates a refusal.
  - `Resource` : Used to describe which resources will be assigned permissions, `*` means for all.
  - `Action` indicates an action that is rejected or allowed, `*` means for all.
    - `s3:GetObject`
    - `s3:GetObjectAcl`
    - `s3:PutObject`
    - `s3:DeleteObject`
    - `s3:PutObjectAcl`
    - `s3:ListMultipartUploadParts`
    - `s3:AbortMultipartUpload`
    - `s3:ListAllMyBuckets`
    - `s3:CreateBucket`
    - `s3:DeleteBucket`
    - `s3:GetBucketVersioning`
    - `s3:GetBucketCORS`
    - `s3:PutBucketAcl`
    - `s3:PutBucketVersioning`
    - `s3:PutBucketCORS`
    - `s3:ListBucket`
    - `s3:ListBucketMultipartUploads`
    - `s3:GetBucketPolicy`
    - `s3:PutBucketPolicy`
    - `s3:DeleteBucketPolicy`
    - `s3:GetBucketAcl`

    Eg: "{\"Statement\":[{\"Resource\":\"*\",\"Action\":[\"s3:*\"],\"Effect\":\"Allow\"}]}"表示允许所有方法访问所有资源。
- RoleArn: Resource name, string type, user defined, length must be greater than 20.
- RoleSessionName: session name, string type, user defined.

Examples of using the Java SDK are as follows:

```
// Generate temporary vouchers from permanent vouchers
String stsEndPoint = "http://sts.ss.bscstorage.com";
String accessKey = "ziwxxxxxxxxxxxxxxxx";
String secretKey = "V+ZTZ5xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx";
BasicAWSCredentials awsCreds = new BasicAWSCredentials(
                accessKey, secretKey);

AWSSecurityTokenService sts = AWSSecurityTokenServiceClientBuilder.standard().
                new AWSStaticCredentialsProvider(awsCreds)).withEndpointConfig
                        new EndpointConfiguration(stsEndPoint, "us-east-1")).b

String policy = "{\"Statement\":[{\"Resource\":\"*\",\"Action\":[\"s3:PutObjec

AssumeRoleRequest assumeRoleRequest = new AssumeRoleRequest();
assumeRoleRequest.setPolicy(policy_str);
assumeRoleRequest.setRoleArn("arn:aws:iam::123456789:role/test_role");
assumeRoleRequest.setRoleSessionName("test-session");

AssumeRoleResult assumeRoleResult = sts.assumeRole(assumeRoleRequest);

String tmp_access_key = assumeRoleResult.getCredentials().getAccessKeyId();
String tmp_secret_key = assumeRoleResult.getCredentials().getSecretAccessKey()
String token = assumeRoleResult.getCredentials().getSessionToken();

// 使用服务器返回的临时凭证
String s3EndPoint = "http://ss.bscstorage.com";

BasicSessionCredentials sessionCredentials = new BasicSessionCredentials(
                tmp_access_key, tmp_secret_key, token);

AmazonS3 s3 = AmazonS3ClientBuilder.standard().withCredentials(
            new AWSStaticCredentialsProvider(sessionCredentials)).withEndpoint
                    new EndpointConfiguration(s3EndPoint, "us-east-1")).build(

S3Object obj = s3.getObject("bucket", "key");
```

Examples of using Python SDK are as follows:

```python
import boto3

cli = boto3.client(
    'sts',
    region_name="bj",
    aws_access_key_id="accesskey",
    aws_secret_access_key="secretkey",
    endpoint_url='http://sts.ss.bscstorage.com'
)

policy = "{\"Statement\":[{\"Resource\":\"*\",\"Action\":[\"s3:*\"],\"Effect\"
# 使用永久凭证生成临时凭证
resp = cli.assume_role(
    DurationSeconds=3600,
    RoleArn='arn:aws:iam::123456789012:role/demo',
    RoleSessionName="test",
    Policy=policy,
)

print("tmp accesskey: %s" % resp['Credentials']['AccessKeyId'])
print("tmp secretkey: %s" % resp['Credentials']['SecretAccessKey'])
print("expired time : %s" % resp['Credentials']['Expiration'])
print("session token: %s" % resp['Credentials']['SessionToken'])

# 使用临时凭证创建client
tmpcli = boto3.client(
    "s3",
    region_name="bj",
    aws_access_key_id=resp['Credentials']['AccessKeyId'],
    aws_secret_access_key=resp['Credentials']['SecretAccessKey'],
    aws_session_token=resp['Credentials']['SessionToken'],
    endpoint_url='http://ss.bscstorage.com',
)

print tmpcli.head_object(Bucket='bucket', Key='key')
```

Examples of using PHP SDK are as follows:

```php
<?php

require 'aws.phar';

try {
    $cli = new Aws\Sts\StsClient([
        'version' => 'latest',
        'region' => 'us-east-1',
        'credentials' => [
            'key' => 'accesskey',
            'secret' => 'secretkey',
        ],
        'endpoint' => 'http://sts.ss.bscstorage.com',
    ]);
    $result = $cli->assumeRole([
        'DurationSeconds' => 3600,
        'Policy' => "{\"Statement\":[{\"Resource\":\"*\",\"Action\":[\"s3:*\"]
        'RoleArn' => 'arn:aws:iam::123456789012:role/demo', // REQUIRED
        'RoleSessionName' => 'test', // REQUIRED
    ]);

    $tmp_accesskey = $result['Credentials']['AccessKeyId'];
    $tmp_secretkey = $result['Credentials']['SecretAccessKey'];
    $tmp_sesstoken = $result['Credentials']['SessionToken'];

    echo "tmp accesskey:", $tmp_accesskey, "\n";
    echo "tmp secretkey:", $tmp_secretkey, "\n";
    echo "session token:", $tmp_sesstoken, "\n";

    $s3 = new Aws\S3\S3Client([
        'version' => 'latest',
        'region' => 'us-east-1',
        'credentials' => [
            'key' => $tmp_accesskey,
            'secret' => $tmp_secretkey,
            'token' => $tmp_sesstoken,
        ],
        'endpoint' => 'http://ss.bscstorage.com',
    ]);

    $resp = $s3->getObject([
        'Bucket' => 'bucket',
        'Key' => 'key',
    ]);

    echo $resp;
} catch (\Exception $e) {
    echo "Exception: " . $e->getMessage() . "\n";
}
```

Examples of using dotnet SDK are as follows:

```
static async Task StsDemo()
{
    AmazonSecurityTokenServiceConfig config = new AmazonSecurityTokenServiceCo
    config.ServiceURL = "http://sts.ss.bscstorage.com";
    config.SignatureVersion = "4";
    AmazonSecurityTokenServiceClient sts = new AmazonSecurityTokenServiceClien
    var response = await sts.AssumeRoleAsync(new AssumeRoleRequest
    {
        Policy = "{\"Statement\":[{\"Resource\":\"*\",\"Action\":[\"s3:*\"],\"
        RoleArn = "arn:aws:iam::123456789012:role/demo",
        RoleSessionName = "testAssumeRoleSession",
    });

    Credentials credentials = response.Credentials;
    Console.WriteLine("tmp accesskey:" + credentials.AccessKeyId);
    Console.WriteLine("tmp secretkey:" + credentials.SecretAccessKey);
    Console.WriteLine("tmp sesstoken:" + credentials.SessionToken);

    AmazonS3Config s3config = new AmazonS3Config();
    s3config.ServiceURL = "http://ss.bscstorage.com";
    s3config.SignatureVersion = "4";
    AmazonS3Client s3 = new AmazonS3Client(credentials.AccessKeyId, credential
    var resp = await s3.GetObjectMetadataAsync("bucket", "key");
    Console.WriteLine(resp.HttpStatusCode);
}
```

Offline Download Interface

Authorization

Use AccessKey and SecretKey.

Please refer to REST-Authentication.

Error Response Status Code and Error

Following is a list of error response definitions, including:

`Status code` `Error code` and its description.

- **400** `InvalidURI`
  Couldn't parse the specified URI.
- **400** `DownloadSourceException`
  Download source exception can not download file.
- **404** `NoSuchDownloadSource`
  The specified download source does not exist.
- **500** `InternalError`
  We encountered an internal error. Please try again.
- **501** `NotImplemented`
  Current request not implemented.

Other status code is same as Amazon S3. Please refer to ErrorResponses.

Signature of all requests is same as Amazon S3.

Commit Offline Download Task

Request URI

`http://offline_domain/<bucket>`

Request Method

PUT

Request Body

Data of JSON format.

Following is a list of request content, including:

Element Name `Element Type` and its description.

- Elements for ALL requests:
  - **Url** `String`
    Download resource URL (including BT, Magnet, HTTP).
  - **SuccessCallbackUrl** `String`
    URL callback success.
  - **FailureCallbackUrl** `String`
    URL callback failure.
  - **Target** `String` *optional*
    Prefix of **Key** for the file to store.
  - **ACLs** `Dict` *optional*
    Access control lists. you can see ACL part in s2 doc for detail

    Examples:

    - Specify the upload file's ACL:

      `task['ACLs'] = {'acl': 'private'}` , `task['ACLs'] = {'acl': 'public-read'}` , `task['ACLs'] = {'acl': 'authenticated-read'}`

    - Authorize the specified user's access to this file:

      ```
      task['ACLs'] = {'grant-read': 'id=user_name', 'grant-write': 'id=user_name', 'grant-read-acp': 'id=user_name', 'grant-write-acp': 'id=user_name'}
      ```

    - Authorize the specified multi users' access to this file:

      `task['ACLs'] = {'grant-full-control': 'id=user_name'}`

      `task['ACLs'] = {'grant-read': 'user_name1', 'grant-write': 'user_name2'}`

      `task['ACLs'] = {'grant-read-acp': 'emailAddress=user_email', 'grant-write-acp': 'id=user_email'}`

- Elements for BT request:
  - **Files** `Array`
    Contains files which user wanted.
- Elements for HTTP request:
  - **Key** `String` *optional*
    Specify put object url, if this field is set, **Target** field will be ignored.
  - **SHA1** `String` *optional*
    SHA1 hash of this file. Check the file's accuracy.
  - **MD5** `String` *optional*
    MD5 hash for this file. Check the file's accuracy.
  - **CRC32** `String` *optional*
    CRC32 hash of this file. Check the file's accuracy.
  - **Size** `Int` *optional*
    Size of this file. Check the file's accuracy.

- **OnKeyExist** `String` *optional*
  specify this file's ignore rule.

  Ignore rules (If the same `key` file exists):

  - `ignore` : Always ignore.
  - `ignore-same-md5` : This key just work with `MD5` key. If this file with the same MD5 value exists, ignore this file.
  - `ignore-same-sha1` : This key just work with `SHA1` key. If this file with the same SHA1 value exists, ignore this file.
  - `ignore-same-crc32` : This key just work with `CRC32` key. If this file with the same CRC32 value exists, ignore this file.
  - `ignore-same-checksum` : This key just work with `MD5` or `SHA1` or `CRC32` key. If this file with the same value of any one of the three exists, ignore this file.
  - `ignore-same-size` : This just work with `Size` key. If this file with the same size exists, ignore this file.
  - `override` : Always override.

Request Example

```
//BT method.
{
    "Files":["movie1.txt","movie2.jpg"],
    "Url":"http://www.abc.com/download/movie.torrent",
    "Target":"movie/2016",
    "SuccessCallbackUrl":"http://website/succ/",
    "FailureCallbackUrl":"http://website/fail/",
}
```

or

```
//HTTP method.
{
    "Url":"http://www.abc.com/download/movie.mp4",
    "Key":"movies/2016/movie.mp4",
    "ACLs": {'acl': 'private'},
    "SHA1":"abcdeabcdeabcdeabcdeabcdeabcdeabcdeabcde",
    "MD5":"abcdabcdabcdabcdabcdabcdabcdabcd",
    "CRC32":"abcdef",
    "SuccessCallbackUrl":"http://website/succ/",
    "FailureCallbackUrl":"http://website/fail/",
}
```

or

```
//Magnet method.
{
    "Url":"magnet:?xt=urn:btih:88594AAACBDE40EF3E2510C47374EC0AA396C08E&dn=mov
    "Target":"movie/2016",
    "SuccessCallbackUrl":"http://website/succ/",
    "FailureCallbackUrl":"http://website/fail/",
}
```

Request Response

- **Success**

  Return status code 200.

- **Failure**

  Same as Amazon, refer to ErrorResponses.

Request Response Header

After request handled successful, it generated a `x-amz-s2-offline-task-id` header, contains this offline download task ID. The subsequent query task and cancel task both can be done by looking up this ID.

Query Status Of Offline Download Task

Request URI

`http://offline_domain/<bucket>/<taskId>`

Request Method

GET

Request Response

- **Success**

  Return status code 200 and body.

  Body contains **State** and **Percent** (total download progress percent). (Json type)

  Example:

  ```
  {
      "State":"TASK_UNDO",
      "Percent":"0.00"
  }
  ```

  or

  ```
  {
      "State":"TASK_DOING",
      "Percent":"80.00"
  }
  ```

  or

```
{
    "State":"TASK_SUCCESS",
    "Percent":"100.00"
}
```

or

```
{
    "State":"TASK_FAILED"
}
```

or

```
{
    "State":"TASK_CANCELLED"
}
```

- **Failure**

  Same as Amazon, refer to ErrorResponses.

Delete Offline Download Task

Request URl

`http://offline_domain/<bucket>/<taskId>`

Request Method

DELETE

Request Response

- **Success**

  Return status code 204.

- **Failure**

  Same as Amazon, refer to ErrorResponses.

notice: current delete download task is not open to user， if you want to use this feature contact us.

Success Callback

Request URL

Determined by argument of `SuccessCallbackUrl` Request.

Request Method

POST

Request Body

Data of JSON format.

Following is a list of request content, including:

Element Name `Element Type` and its description.

Not null for ALL of these elements.

- **ETag** `String`
  MD5 hash of this file. This md5 value must be enclosed in double quotes.
- **MD5** `String`
  MD5 hash of this file.
- **SHA1** `String`
  SHA1 hash of this file.
- **CRC32** `String`
  CRC32 hash of this file.
- **Key** `String`
  Name of this file. (e.g. `<Target>/<File> or <Key>`)
- **Size** `Int`
  Size of this file.
- **Content-Type** `String`
  MIME type of this file (Converted to lowercase letters)
- **LastModified** `String`
  Upload time. (Unit: ms)
- **Task** `String`
  Name of this offline download task. (e.g. `<bucket>/<taskId>`)
- **Concrete-Upload** `bool`
  Concrete upload file to storage, (offline will ignore upload file if it already existed in storage)
- **Success-IDC** `String`
  file has uploaded succeed idc list, use comma to split idcs. if Concrete-Upload value is false, this field is null
- **Failure-IDC** `String`
  file has uploaded failed idc list, use comma to split idcs. if Concrete-Upload value is false, this field is null

Request Example

```
{
    "movie1":{
        "ETag":"\"abcdabcdabcdabcdabcdabcdabcdabcd\"",
        "MD5":"abcdabcdabcdabcdabcdabcdabcdabcd",
        "SHA1":"abcdeabcdeabcdeabcdeabcdeabcdeabcdeabcde",
        "CRC32":"fasdfasdf",
        "Key":"offline/prefix/movie1",
        "Size":"123456",
        "Content-Type":"video\/mp4",
        "LastModified":"1467355438043.852",
        "Task":"bucket/0000d1f0010000000062",
        "Concrete-Upload": true,
        "Success-IDC": "idc1,idc2",
        "Failure-IDC": ""
    } ,
}

or

{
    "movie1":{
        "ETag":"\"abcdabcdabcdabcdabcdabcdabcdabcd\"",
        "MD5":"abcdabcdabcdabcdabcdabcdabcdabcd",
        "SHA1":"abcdeabcdeabcdeabcdeabcdeabcdeabcdeabcde",
        "CRC32":"fasdfasdf",
        "Key":"offline/prefix/movie1",
        "Size":"123456",
        "Content-Type":"video\/mp4",
        "LastModified":"1467355438043.852",
        "Task":"bucket/0000d1f0010000000062",
        "Concrete-Upload": false,
        "Success-IDC": null,
        "Failure-IDC": null
    } ,
}
```

Request Response

- **Success**

  Return status code 200.

- **Failure**

  Same as Amazon, refer to ErrorResponses.

Failure Callback

Request URL

Determined by argument of FailureCallbackUrl Response.

Request Method

POST

Request Body

Data of JSON format.

Following is a list of request content, including:

Element Name `Element Type` and its description.

Not null for ALL of these elements EXCEPT **Files**.

- **Code** `String`
  Error code.
- **Message** `String`
  Error description information.
- **Resource** `String`
  Resource name, contains URL of this offline request task.
- **RequestId** `String`
  Request ID.
- **Task** `String`
  Name of this offline download task. (e.g. `<bucket>/<taskId>`)
- **Files** `dict` *optional*
  Contains information of files which has been uploaded. With this element, users can pick up information of files which has been uploaded sucessful.

Request Example

```
{
    "Code":"NoSuchDownloadSource",
    "Message":"The specified download source does not exist",
    "Resource":"http://www.abc.com/download/movie.torrent",
    "RequestId":"4442587FB7D0A2F9",
    "Task":"bucket/0000d1f0010000000062",
}
```

or

```
{
    "Code":"InternalError",
    "Message":"We encountered an internal error. Please try again.",
    "Resource":"http://www.abc.com/download/movie.torrent",
    "RequestId":"4442587FB7D0A2F9",
    "Task":"bucket/0000d1f0010000000063",
    "Uploaded_Files": {
        "movie1": {
            "ETag":"\"abcdabcdabcdabcdabcdabcdabcdabcd\"",
            "SHA1":"abcdeabcdeabcdeabcdeabcdeabcdeabcdeabcde",
            "CRC32":"fasdfasdf",
            "Key":"offline/prefix/movie1",
            "Size":"123456",
            "Content-Type":"video\/mp4",
            "LastModified":"1467355438043.852",
        }
    }
}
```

Request Response

- **Success**

  Return status code 200.

- **Failure**

  Same as Amazon, refer to ErrorResponses.

The Description of Imgx

Tag： Image process Imgx BaishanCloud Cloud Storage

URL Format

```
[http://imgx-ss.bscstorage.com/<bucket>/ <processing](http://imgx-ss.bscstorag
```

OR

```
[http://<bucket>.imgx-ss.bscstorage.com/<processing]() instruction>/<The File
```

- `bucket`: Your bucket name in Baishan Cloud storage.
- `Processing Instruction`: The processing instruction for the original image. There are some detailed instructions below.
- `文件路径`: The storage path of the original image.
- `Signature`: The main parameters of the Signature.

Example

Example: The URL of the original image is http://ss.bscstorage.com/imgx-test/demo/1.jpg?AWSAccessKeyId=acc_drdrxp&Expires=2464773061&Signature=jFVHRSrOLeg5e3nIR00UE2vik0A%3D It shows

`` `- The bucket name is imgx-test `- The storage path of the bucket is demo/1.jpg

Operations:

`` `- Crop the face area and shorten the image to 400x400 - c *thumb,g* face,w *400,h* 400 ` - Turn the image into a rounded corner with the maximum radius (perfect circle) - `r_ max` `- Increase the brightness of the image by 8%` e *brightness - 8* ` *- Convert the image format to PNG - `f* png`

The Processing Instruction: `c_ thumb,g_ face,w_ 400,h_ 400,r_ max,e_ brightness:8,f_ png` We can access it directly through the URL as below

```
http://imgx-ss.bscstorage.com/imgx-test/c_
thumb,g_
face,w_
400,h_
400,r_
max,e_
brightness%3A8,f_
png/demo/1.jpg?AWSAccessKeyId=acc_
drdrxp&Expires=2464773122&Signature=4cwxt1e%2Fg2NL1OfsldaVsa8SD9s%3D
```

OR

```
http://imgx-test.imgx-ss.bscstorage.com/c_
thumb,g_
face,w_
400,h_
400,r_
max,e_
brightness%3A8,f_
png/demo/1.jpg?AWSAccessKeyId=acc_
drdrxp&Expires=2464773122&Signature=4cwxt1e%2Fg2NL1OfsldaVsa8SD9s%3D
```

Or you could create a Jason file if you do not want to expose the processing instruction in the URL.

```
[
{
"crop" : "thumb",
"gravity" : "face",
"width" : 400,
"height" : 400,
"radius" : "max",
"effect" : "brightness:8",
"format" : "png"
}
]
```

Save the File to the path which belongs to your bucket.

```
<p>imgx/cmd_
template/my_
thumb.json    </p><p>#The ‘my_
thumb’ is a custom name by yourself.</p>
```

Then the user could access it through the below URL which means the processing instruction can be hidden to the Jason File.

```
http://imgx-ss.bscstorage.com/imgx-test/t_
my_
thumb/demo/1.jpg?AWSAccessKeyId=acc_
drdrxp&Expires=2464773563&Signature=WIMaNLacGaPRAlA6fl%2BGAsStfoQ%3D
```

Processing Format

- The connection between the Processing name and the Processing value is '_' such as `c_fit` .
- Related instructions are separated by commas such as `c_fit,w_100,h_100,g_face` .
- Different instructions are separated by '--', such as `c_fit,w_100,h_100,g_face,r_max--l_text:my_font:hello+word,w_100,h_40`

Signature

To protect your original image from theft and processing instructions from malicious abuse, signature protection must be used here. The signature form is fully compatible with AWS V2 authentication. The following provides a PHP function to generate the URL of the generation signature:

```
/**
* Get a query string authenticated URL
*
* @param string $accessKey AccessKey
* @param string $secretKey SecretKey
* @param string $bucket Bucket name
* @param string $uri Object URI
* @param integer $lifetime Lifetime in seconds
* @param boolean $hostBucket Use the bucket name as the hostname
* @param boolean $https Use HTTPS ($hostBucket should be false for SSL verific
* @return string
*/
function getAuthenticatedURL($accessKey, $secretKey, $bucket, $uri, $lifetime,

$expires = time() + $lifetime;
$uri = str_
replace(array('%2F', '%2B', '%2C', '%3A', '%20'), array('/', '+', ',', ':', '+
return sprintf(($https ? 'https' : 'http').'://%s/%s?AWSAccessKeyId=%s&Expires
$hostBucket ? $bucket : $endpoint.'/'.$bucket, $uri, $accessKey, $expires,
urlencode(base64_
encode(hash_
hmac('sha1', "GET\n\n\n{$expires}\n/{$bucket}/{$uri}", $secretKey, true))));
}
//The usage method
echo getAuthenticatedURL('your accessKey', 'your secretKey', 'your bucket', 't
# We recommend to set the value of $lifetime more longer, such as 1000000000.
```

Cache & CDN

- The cached will be generated automatically after processing the image and the same request will not be generated. The default expired time is 2 days.
- If the user configures the CDN address, the processed image will be automatically sent to the distributed nodes.

The Processing Instructions

> The processing instructions are introduced below, and the following original pictures are used as examples for your comparison.
>
> 1. demo/charles.png
> 2. demo/1.jpg
> 3. demo/3.png
> 4. demo/4.png
> 5. demo/sheep.png
> 6. demo/horses.png
> 7. avatar.png

Processing instruction name	*processing instruction*	*Value*	*example*	
crop	**c**	*mode*		**C**
		scale		Ch the of ma c_
		fill		Cr ori c_
		lfill		Th dif to c_
		fit		Ch the ret pa Eq du c_
		mfit		Th dif to c_
		limit		Th dif pr im c_
		pad		Sp ret sc wil c_
		lpad		Th dif lar no c_
		mpad		Th dif to c_

Processing instruction name	*processing instruction*	*Value*	*example*	
crop	**c**	***mode***		**C**
		crop		Sp cr im c_
		thumb		To 'fa ge to c_
width	**w**	***Pixels or percentages***		
		80		Ad w_
		0.1		Ad siz w_
height	**h**	***Pixels or percentages***		**H**
		80		Ad h_
		0.1		Ad siz h_
gravity	**g**	***To specify a position or direction***		**2.**
		north_west		Up c_
		north		St c_
		north_east		Up c_
		west		Or c_
		center		Th c_

Processing instruction name	*processing instruction*	*Value*	*example*	
crop	**c**	***mode***		**C**
		east		Or c_
		south_west		Lo c_
		south		Str c_
		south_east		Lo c_
		xy_center		Th the c h_4
		face		Au mu ea c_
		face (thumb)		Au fac to fac rec c_
		faces		Au mu c e_t
		face:center		Au fac au the c_
		faces:center		Au mu fou ce c h_1
x	**x**	***Pixels***		

Processing instruction name	*processing instruction*	*Value*	*example*	
crop	**c**	***mode***		**C**
		110		Cr sta c_
y	**y**	***Pixels***		**To**
		230		Cr sta c_
quality	**q**	***percentage***		**C JI va**
		100		Im 14 c_
		10		Im rec c_
radius	**r**	***Pixel or 'Max'***		
		30		Ge rad c_
		max		Us rou c_
angle	**a**	***Angle or flip mode***		
		90		Ro c_
		10		Ro c_
		-20		Ro c_
		vflip		flip c_

Processing instruction name	*processing instruction*	*Value*	*example*	
crop	**c**	***mode***		**C**
		hflip		flip c_
effect	**e**	***name and value***		
		grayscale		gra c_
		oil_ paint		Ca c_
		oil_paint:2		Us lev to c_
		negate		rev c_
		brightness:28		Ad sp va de c_
		brightness:-28		Ad sp va de c_
		blur		blu c_
		blur:300		Us va 20 c_
		pixelate		pix c_
		pixelate:40		Us lev c_
		sharpen		Sh c_

Processing instruction name	*processing instruction*	*Value*	*example*	
crop	**c**	***mode***		**C**
		sharpen:400		Us va 20 c_
		auto_ contrast		Au c_
		improve		Au co c_
		sepia		Ad eff c_
		sepia:60		Ad eff Th de c_
		red:40		Ad c_
		green:40		Ad c_
		blue:40		Ad c_
		yellow:40		Ad c_
		cyan:40		Ad c_
		magenta:40		Ad c_
opacity	**o**	***Percentage***		**Co in**
		25		Th h_
border	**bo**	***style***		

Processing instruction name	*processing instruction*	*Value*	*example*	
crop	**c**	***mode***		**C**
		10_ 0000004a		Se bla `h_`
		8_bbbbbb		Se va `h_`
background	**b**	***color***		**S**
		dddddd		Se va `c_`
		fff6def0		Se op `c_`
		dbeced		Se va `c c_`
overlay	**l**	***Use watermark picture name, or use font to describe the file name, etc***		**in Y v**
		superman		Ad su the Fir ma co im yo to `c l s w l_s`
		text:font_me:Hello, BaishanCloud		Te Se `l g_`
format	**f**	***Image Format***		
		png		`f_`

Processing instruction name	*processing instruction*	*Value*	*example*	
crop	**c**	***mode***		**C**
		webp		f_
		jpeg		f_
		jpg		f_
version	**v**	***Version***		**S**
		1.21		de
		13		int
transformation	**t**	***Name***		**"Ir b j**
		my_thumbs		Cu bu file t_
information	**i**	***information type***		**R**
		exif	exif	i_e im
		iptc	iptc	i_i the
		all	all	i_a im

Access the image processing service by using AWS-SDK

You can access the image processing service directly using the standard AWS-SDK.

Note: The IMGx service only accepts the standard 'get_ object' operation. All other operations are illegal.

The following example code uses python boto3 sdk：

```
import boto3

s2_imgx_domain_name = 'http://imgx-ss.bscstorage.com'   # This example uses th
s2_imgx_access_key = 'xxxxxx'                           # User own access key
s2_imgx_secret_key = 'xxxxxx'                           # User own secret key

imgx_cmd = 'c_scale,w_100,h_100' # Picture processing commands, refer to the p
imgx_bucket = 'my_imgx_bucket'   # The name of the user's bucket on the storag
imgx_key = 'my_img.jpg'          # The key name of the image that the user wan


# Create s2 client
imgx_cli = boto3.client('s3',
endpoint_url=s2_imgx_domain_name,
aws_access_key_id=s2_imgx_access_key,
aws_secret_access_key=s2_imgx_secret_key)

imgx_cmd_key = u'{cmd}/{key}'.format(cmd=imgx_cmd, key=imgx_
key) # As the Key on final access
resp = imgx_cli.get_object(Bucket=imgx_bucket, Key=imgx_
cmd_key)

print resp
```

Instruction about the watermark function

1. WaterMark type: `picture watermarking` and `text watermarking`.
2. If you use the watermark `l` command, other instructions will be combined with `l` to process the `watermark image or text` (e.g. `w`、`h`、`g`、`x`、`y`、`o`、`bo`、`e`、`a`、`r`, etc.)

Picture Watermark

You need to save the watermark texture to the corresponding bucket `imgx/overlay/<filename>.png` in advance. The image must be in png format. The following two images are examples.

watermark texture	***corresponding path***	***corresponding command***
	*imgx/overlay/icon_v.png*	`l_icon_v`
	*imgx/overlay/bs_logo.png*	`l_bs_logo`

The following are examples of specific functions

example	*description*	*processing instruction*
	1. Scale the original image proportionally; 2. Add an image watermark to the upper left corner of the original image, fine-tune the coordinates (x_43, y_20), set the width of the watermark to 120px, the opacity to 35%, and rotate the watermark 10 degrees counterclockwise.	`c_fit,w_300,f_png--l_bs_logo,g_north_west,w_120,o_35,x_43,y_20,a_-10`
	1. Process the original image into a rounded avatar 2. Add a watermark image in the lower right corner, fine-tune the watermark coordinates (x_-1, y_-5) outward, and set the width of the watermark image to 60px.	`c_thumb,g_face,w_200,h_200,r_max,bo_6_ffffff80,f_png--l_icon_v,g_south_east,w_60,x_-1,y_-5`
	1. As above 2. Invert the watermarked image	`c_thumb,g_face,w_200,h_200,r_max,bo_6_ffffff80,f_png--l_icon_v,g_south_east,w_60,x_-1,y_-5,e_negate`

Text Watermark

Please save the font configuration of the text watermark (json formatted file) to the corresponding bucket `imgx/overlay/<file>.json` in advance, for example,

```
{
"font_family" : "Xingkai SC",
"font_style" : "bold",
"font_size" : 30,
"font_color" : "000000",
"background" : "ff0000cc",
"padding" : 10,
"word_spacing" : 1,
"kerning" : 1,
"line_spacing" : 2,
"pierced" : false,
"tile" : false,
"text" : "Default Value",
}
```

Corresponding path: `imgx/overlay/my_font.json`

Introduction of font parameters

All parameters are not required

parameter name	*description*	*default value*
font_family	All fonts we support	Songti SC
font_style	Font style (only effective if the font itself supports the following styles, otherwise normal is used by default): normal（normal） *italic*（italic face） **bold**（bold face） light（light face type）	normal
font_size	font size（px）	14
font_color	Hexadecimal rgba (the first 6 are rgb, the last 2 bits of alpha are 0 to f), if the last two digits are omitted, it is considered opaque	Black(000000)
background	16Hexadecimal rgba (the first 6 are rgb, the last 2 bits of alpha are 0 to f), if the last two digits are omitted, it is considered opaque	transparent(ffffff00)
padding	The width of the padding around the text（px）	6
word_spacing	character spacing（px）	0
kerning	word spacing（px）	0
line_spacing	Line spacing（px）	0
pierced	cutout effect（bool value）	false
tile	Whether to tile（bool value）	false
text	default string	empty string

Example 1

`imgx/overlay/simple_font.json` :

```
{
"font_family" : "Microsoft YaHei",
"font_size" : 40,
"font_color" : "ffffff"
}
```

The simplest text watermarks 

Processing Instruction:

```
w_800,f_png--l_text:simple_font:Hello+Bai+Shan!!,x_20,y_20,a_-25
```

Example 2

`imgx/overlay/subtitles.json` :

```
{
"font_size" : 22,
"font_color" : "ffffff"
}
```

`imgx/overlay/subtitles_s.json` :

```
{
"font_size" : 16,
"font_color" : "ffffff"
}
```

Let's create a blockbuster effect and apply it to two images: 

Processing Instruction:

```
f_png,c_fill,w_800,h_400,e_brightness:-8--c_pad,w_800,h_550,g_center,b_000000f
100--l_text:subtitles_s:Imgx is the image back-end for web and mobile develope
```

Example 3

`imgx/overlay/my_font.json` :

```
{
"font_family" : "Xingkai SC",
"font_style" : "bold",
"font_size" : 40,
"font_color" : "000000",
"background" : "ff0000cc",
"padding" : 18
}
```

Let's make a couplet: 

Processing Instruction:

```
w_800,f_png--l_text:my_font,g_south_west,w_40,x_20,y_100--l_text:my_font,g_sou
```

Example 4

`imgx/overlay/font_me.json` :

```
{
"font_family": "Microsoft YaHei",
"font_size": 30,
"font_color": "ffffff",
"font_style": "bold",
"background": "0000008f",
"pierced": true,
"tile": false,
"padding": 12,
"word_spacing": 2.2,
"line_spacing": 1.2,
"kerning": 1.5
}
```

Cutout Effect 

指令：

```
l_text:font_me: Hello,BaishanCloud,g_north_west,x_20,y_20--w_800
```

Example 5

`imgx/overlay/tile.json` ：

```
{
"font_family" : "Microsoft YaHei",
"font_style" : "bold",
"font_size" : 40,
"font_color" : "000000",
"background" : "ff000066",
"padding" : 18,
"tile": true
}
```

tiling effect 

Processing Instruction:

```
w_800,f_png--l_text:tile:Hello BaiShan!!,g_south
```

Example 6

`imgx/overlay/badge.json` ：

```
{
"font_style" : "bold",
"font_size" : 30,
"font_color" : "ffffff",
"background" : "ff0000cc",
"padding" : 15
}
```

Add a badge 

Processing Instruction:

```
c_thumb,g_face,w_200,h_200,r_max,bo_6_ffffff80,f_png--l_text:badge:69,r_max,g_
```

Use the "instruction set"

If you don't want to expose the directive in the URL, this is an example to automatically recognize the face position on a photo and make a circular thumbnail with a border:

- Firstly, create a json file and save it to the corresponding bucket: Path: `imgx/cmd_template/avatar.json` :

```
[
{
"crop" : "thumb",
"gravity" : "face:center",
"width" : 200,
"height" : 200,
"radius" : "max",
"border" : "6_ffffff80",
"format" : "png"
},
{
"overlay" : "icon_v",
"gravity" : "south_east",
"width" : 60,
"x" : -1,
"y" : -5
}
]
```

- Access via the below way.

```
http://imgx-ss.bscstorage.com/imgx-test/t_avatar/demo/1.jpg?<signature>
http://imgx-ss.bscstorage.com/imgx-test/t_avatar/demo/3.jpg?<signature>
```

- Effect： 

Font (font_family)

> The API to query fonts: http://imgx-ss.bscstorage.com/fonts

How to use video and audio transcoding service

Function list

Supported input formats

- container format

3GP、AVI、FLV、MP4、M3U8、MPG、WMV、MKV、MOV、TS、 DIV、GIF、AMR、MP3

- Video coding format

H.264、H.265、MPEG-1、MPEG-2、MPEG-4、VP8、VP9、Windows Media Video等

- Audio coding format

AAC、AC-3、MP2、MP3、SPEEX、Windows Media Audio

Supported output formats

- Container format

FLV、MP4、TS、M3U8、MPG、TS、GIF、MP3、MP2、AAC、WMV、WMA、AAC、DASH

- Video coding format

H.264、H.265、GIF、MPEG-2、MS-MPEG4

- Audio coding format

AAC、MP2、MP3、Windows Media Video

- Encoding format compatibility relationship

Coding format	Supported containers
H.264	FLV, MP4, TS
H.265	FLV, MP4, TS
GIF	GIF
MPEG2	MPG
AAC	FLV, MP4, TS, AAC
MP2	MPG, MP2
MP3	MP3, FLV, MP4, TS
MP3	MP3, FLV, MP4, TS
MS-MPEG4	WMV
Windows Media Audio	WMV, WMA

Transform method

- Auto transcoding:

Automatically transcode video files uploaded to a bucket by configuring transcoding templates and transcoding rules in advance

- Active transcoding:

Users actively transcode a single file by creating transcoding tasks (transcoding templates and pipelines need to be created in advance)

Transcoding function

- Video screenshot: capture JPG / PNG pictures at a certain time point or at a time interval of the video file.
- Video clip: capture a period of time in the video for transcoding. (under development)
- Rotational resolution
- To video bit rate
- To audio bit rate
- To audio sampling rate
- To video frame rate
- Set GOP (keyframe fixed spacing)
- HLS transcoding
- DASH transcoding
- FastStart：The fast start of playback is realized by moving the original information of MP4 file from the end of the file to the head.

Noun interpretation

Transcoding template（Preset）

Transcoding templates are used to predefine transcoding rules that can be reused.

Task（Job）

The task is responsible for transcoding the multimedia files stored in the bucket according to the pre created template. A transcoding task can transcode an output into multiple output formats. To improve parallel efficiency, it is recommended to create a task for each output.

Pipeline

A pipeline is a queue used to manage tasks. When creating a task, you need to select a pipe to store the task to be created. You can create multiple pipes to perform different tasks. For example, you can create two pipelines, one for ordinary tasks and the other for urgent tasks only. The tasks in each pipeline can be executed concurrently, but the number of concurrent tasks is limited (currently 5 tasks). The transcoding system executes in turn according to the order in which the tasks in each pipeline are created.

How to use transcoding system

Create transcoding template

In the Storage Console http://cwn-ss.bscstorage.com/#/video/template Module, create a transcoding template through the UI.

Create pipe

use https://doc.bscstorage.com/doc/transcoder/apis/pipeline/create_pipeline.html Create a pipe.。

Create task

use https://doc.bscstorage.com/doc/transcoder/apis/job/create_job.html Create a pipe.

Transcoder Request signature method

All requests in the transcoder API require signature verification.Transcoder service currently supports AWS V4 signature. It includes the following four steps:

1.establish Canonical Request

2.establish String to Sign （Please pay attention to the use elastictranscoder service name ）

3.Generate a signature using the HMAC method.

For specific V4 signature steps, please refer to: signature algorithm

Create pipe

Description

Send post request to `/2012-09-25/pipelines/` to create a pipe.

request

```
POST /2012-09-25/pipelines HTTP/1.1
Content-Type: application/json; charset=UTF-8
Accept: */*
Host: transcoder-ss.bscstorage.com
x-amz-date: 20170726T174952Z
Authorization: AWS4-HMAC-SHA256
               Credential=AccessKeyID/request-date/Transcoder endpoint/elastic
               SignedHeaders=host;x-amz-date;x-amz-target,
               Signature=calculated-signature
Content-Length: number of characters in the JSON string
{
   "Name":" 管道名称",
   "InputBucket":"用于存放转码源文件的 Bucket 名称",
   "OutputBucket":"用于存放输出文件的 Bucket 名称",
   "ContentConfig":{
      "Permissions":[
         {
            "GranteeType":"Canonical|Email|Group",
            "Grantee":"用户名(Canonical)|用户邮箱(Email)|
                      AllUsers|AuthenticatedUsers|LogDelivery(Group)",
            "Access":[
               "Read|ReadAcp|WriteAcp|FullControl",
               ...
            ]
         },
         {...}
      ],
   },
   "SuccessCallbackUrl": "接受任务成功回调的 URL",
   "FailureCallbackUrl": "接受任务失败回调的 URL",
}
```

Request parameters

- Name

Pipe name. In order to facilitate your management, we recommend setting different names for different pipes, but it is not forced. The maximum length of the pipe name is 40 characters.

- InputBucket

The name of the bucket used to store the transcoding source file. Please confirm that the entered bucket exists in your account.

- OutputBucket

The bucket name used to store the transcoded output file. Please confirm that the entered bucket exists in your account.

- (Optional) ContentConfig: Permissions

Defines which users or predefined user groups can access or modify the transcoded input file. If you set permissions, the system will only grant you the specified permissions to the transcoded files, and will not grant full control permissions to the owner. If you ignore this item, the system grants full control to the owner.

- (Optional) SuccessCallbackUrl

Accept the URL of the successful callback of the task. Only HTTP protocol URLs are supported. After the transcoding task is successful, the system will send a post request to the URL. The request JSON content is as follows:

```
{
    "result": "success",
    "input_bucket": "in_bucket_name",
    "input_key": "test.mp4",
    "output_bucket": "out_bucket_name",
    "output_keys": ["test_300K.mp4",

    # HLS 转码相关
    "output_playlists": ["test_300K.m3u8"],

    # 转码后输出文件元信息
    "metadata": {
        "test_300K.mp4": {
            'etag': "xxxx",
            'video_streams': [
                {
                    "index": 0,
                    "codec_name": "h264",
                    "width": 560,
                    "height": 320,
                    "duration_ts": 498000,
                    "duration": "5.533333",
                    "bit_rate": "300000",
                    ...
                }, {}, ...
            ],
            'audio_stream': [
                {
                    "index": 1,
                    "codec_name": "aac",
                    "sample_fmt": "fltp",
                    "sample_rate": "48000",
                    "channels": 1,
                    "channel_layout": "mono",
                    "bits_per_sample": 0,
                    "bit_rate": "83050",
                    ...
                }
            ],
            'format': {
                "nb_streams": 2,
                "nb_programs": 0,
                "format_name": "mov,mp4,m4a,3gp,3g2,mj2",
                "format_long_name": "QuickTime / MOV",
                "start_time": "0.000000",
                "duration": "5.568000",
                "size": "383631",
                "bit_rate": "323000",
            }
        },
        "test_300K.m3u8: { ... }
    }
}
```

- (optional) FailureCallbackUrl

Accept the URL of the successful callback of the task. Only HTTP protocol URLs are supported. After the transcoding task fails, the system will send a post request to the URL. The request JSON content is as follows:

```
{
    "result": "failure",
    "input_bucket": "in_bucket_name",
    "input_key": "test.mp4",
    "error_type": "InternalError",
    "error_message": "...",
}
```

Return

```
Status: 201 Created
Content-Type: application/json
Content-Length: number of characters in the response
Date: Mon, 14 Jun 2017 06:01:47 GMT

{
   "Pipeline":{
       "Id":"1234567890123456 78",
       "Name":" my_pipeline",
       "InputBucket":"input_bucket",
       "OutputBucket":"output_bucket",
       "ContentConfig":{
          "Permissions":[
             {
                "GranteeType":"Group",
                "Grantee":"AllUsers",
                "Access":["Read"]
             }
          ],
       },
       "SuccessCallbackUrl": "http://mydomain.com/cb",
       "FailureCallbackUrl": "http://mydomain.com/cb",
       "Status": "Active|Paused",
    }
}
```

Get pipe

Description

Send GET request to `/2012-09-25/pipelines/` to get the pipeline belonging to your account.

request

```
GET /2012-09-25/pipelines/pipelineId HTTP/1.1
Content-Type: charset=UTF-8
Accept: */*
Host: transcoder-ss.bscstorage.com
x-amz-date: 20170726T174952Z
Authorization: AWS4-HMAC-SHA256
               Credential=AccessKeyID/request-date/Transcoder endpoint/elastic
               SignedHeaders=host;x-amz-date;x-amz-target,
               Signature=calculated-signature
```

return

```
Status: 200 OK
Content-Type: application/json
Content-Length: number of characters in the response
Date: Mon, 14 Jun 2017 06:01:47 GMT

{
   "Pipeline":{
       "Id":"123456789012345678",
       "Name":" my_pipeline",
       "InputBucket":"input_bucket",
       "OutputBucket":"output_bucket",
       "ContentConfig":{
          "Permissions":[
             {
                "GranteeType":"Group",
                "Grantee":"AllUsers",
                "Access":["Read"]
             }
          ],
       },
       "SuccessCallbackUrl": "http://mydomain.com/cb",
       "FailureCallbackUrl": "http://mydomain.com/cb",
       "Status": "Active|Paused",
    }
}
```

View pipe

Description

Send GET request to `/2012-09-25/pipelines/pipelineId` to get the specified pipe.

request

```
GET /2012-09-25/pipelines?PageToken=value HTTP/1.1
Content-Type: charset=UTF-8
Accept: */*
Host: transcoder-ss.bscstorage.com
x-amz-date: 20170726T174952Z
Authorization: AWS4-HMAC-SHA256
               Credential=AccessKeyID/request-date/Transcoder endpoint/elastic
               SignedHeaders=host;x-amz-date;x-amz-target,
               Signature=calculated-signature
```

- PageToken

When the transcoder interface returns more than one page of results (`NextPageToken` appears in the return) Please use `PageToken` to get the results on the next page.

return

```
Status: 200 OK
Content-Type: application/json
Content-Length: number of characters in the response
Date: Mon, 14 Jun 2017 06:01:47 GMT

{
   "Pipelines": [
       {
           "Id":"123456789012345678"
           "Name":" my_pipeline",
           "InputBucket":"input_bucket",
           "OutputBucket":"output_bucket",
           "SuccessCallbackUrl": "http://mydomain.com/cb",
           "FailureCallbackUrl": "http://mydomain.com/cb",
           "ContentConfig": {...},
           "Status": "Active|Paused",
        },
        { ... }
    ],
    "NextPageToken": "123456789012345679"
}
```

Delete pipe

Description

Send DELETE request to `/2012-09-25/pipelines/pipelineId` to get the specified pipe.

Request

```
DELETE /2012-09-25/pipelines/pipelineId HTTP/1.1
Content-Type: charset=UTF-8
Accept: */*
Host: transcoder-ss.bscstorage.com
x-amz-date: 20170726T174952Z
Authorization: AWS4-HMAC-SHA256
               Credential=AccessKeyID/request-date/Transcoder endpoint/elastic
               SignedHeaders=host;x-amz-date;x-amz-target,
               Signature=calculated-signature
```

Return

```
Status: 202 Accepted
Content-Type: application/json
Content-Length: number of characters in the response
Date: Mon, 14 Jun 2017 06:01:47 GMT

{
   "Success":"true"
}
```

Update pipeline

Describe

Send PUT request to `/2012-09-25/pipelines/pipelineId` update the created pipe.

How to update queue parameters

Please include all updated queue parameters in the requested body (unchanged parameters also need to be included).

Request

```
PUT /2012-09-25/pipelines/pipelineId HTTP/1.1
Content-Type: application/json; charset=UTF-8
Accept: */*
Host: transcoder-ss.bscstorage.com
x-amz-date: 20170726T174952Z
Authorization: AWS4-HMAC-SHA256
               Credential=AccessKeyID/request-date/Transcoder endpoint/elastic
               SignedHeaders=host;x-amz-date;x-amz-target,
               Signature=calculated-signature
Content-Length: number of characters in the JSON string
{
   "Name":" 管道名称",
   "InputBucket":"The Bucket name used to store the transcoding source file",
   "OutputBucket":"The name of the Bucket used to store the output file",
   "ContentConfig":{
      "Permissions":[
         {
            "GranteeType":"Canonical|Email|Group",
            "Grantee":"用户名(Canonical)|用户邮箱(Email)|
                      AllUsers|AuthenticatedUsers|LogDelivery(Group)",
            "Access":[
               "Read|ReadAcp|WriteAcp|FullControl",
               ...
            ]
         },
         {...}
      ],
   },
   "SuccessCallbackUrl": "Accept successful callback of task URL",
   "FailureCallbackUrl": "Accept callback of task failure URL",
}
```

Request parameters

- Name

Pipe name. In order to facilitate your management, we recommend setting different names for different pipes, but it is not forced. The maximum length of the pipe name is 40 characters.

- InputBucket

The name of the bucket used to store the transcoding source file. Please confirm that the entered bucket exists in your account.

- OutputBucket

The bucket name used to store the transcoded output file. Please confirm that the entered bucket exists in your account.

- (Optional) ContentConfig: Permissions

Defines which users or predefined user groups can access or modify the transcoded input file. If you set permissions, the system will only grant you the specified permissions to the transcoded files, and will not grant full control permissions to the owner. If you ignore this item, the system grants full control to the owner.

- (Optional) SuccessCallbackUrl

Accept the URL of the successful callback of the task. Only HTTP protocol URLs are supported. After the transcoding task is successful, the system will send a post request to the URL.

- (Optional) FailureCallbackUrl

Accept the URL of the successful callback of the task. Only HTTP protocol URLs are supported. After the transcoding task fails, the system will send a post request to the URL.

return

```
Status: 202 Accepted
Content-Type: application/json
Content-Length: number of characters in the response
Date: Mon, 14 Jun 2017 06:01:47 GMT

{
   "Pipeline":{
       "Id":"123456789012345678",
       "Name":" my_pipeline",
       "InputBucket":"input_bucket",
       "OutputBucket":"output_bucket",
       "SuccessCallbackUrl": "http://mydomain.com/cb",
       "FailureCallbackUrl": "http://mydomain.com/cb",
       "ContentConfig": {...},
       "Status": "Active|Paused",
    }
}
```

Create task

Description

Send POST request to `/2012-09-25/jobs` to create a task.

Request

```
POST /2012-09-25/jobs HTTP/1.1
Content-Type: application/json; charset=UTF-8
Accept: */*
Host: transcoder-ss.bscstorage.com
x-amz-date: 20170726T174952Z
Authorization: AWS4-HMAC-SHA256
               Credential=AccessKeyID/request-date/Transcoder endpoint/elastic
               SignedHeaders=host;x-amz-date;x-amz-target,
               Signature=calculated-signature
Content-Length: number of characters in the JSON string
{
   "Inputs":[{
      "Key":"转码源文件名"
   }],
   "OutputKeyPrefix":"转码后的输出文件名前缀",
   "Outputs":[{
      "Key":"转码后的输出文件名",
      "PresetId":"转码模板 ID",
      "SegmentDuration":"[1,60]",
      "Watermarks": [{
        "InputKey": "string",
      }],
      "Encryption": {
        "Mode": "string",
        "Key": "string",
        "KeyMd5": "string",
        "InitializationVector": "string"
      },
      "Composition": [{
        "TimeSpan": {
          "StartTime": "string",
          "Duration": "string"
        }
      }],
   },
   {...}],
   "Snapshots":[{
       "Key":"转码后的输出文件名",
       "Format":"jpg|png",
       "Time":"截图开始时间点",
       "Interval":"截图间隔时间",
       "Number":"截图数量",
   },
   {...}]
   "Playlists":[{
      "Format":"HLSv3|HLSv4",
      "Name":"播放列表文件名",
      "OutputKeys":[
         "包含在输出文件列表中的输出文件名",
         ...
      ],
   },
   {...}],
   "PipelineId":"本转码任务使用的管道 ID"
}
```

Request parameters

- Inputs

Transcoding source file information. Accept type: array. Note that currently only a single input is supported. For example:

```
"Inputs": [{
    "Key": "path/to/orginal.mp4"
}]
```

- Inputs: Key

Transcoding the original file name.

- OutputKeyPrefix

Output file name prefix after transcoding. It is usually used to store transcoded output in a separate directory. Such as setting OutputKeyPrefix is transcode_output/, So that the transcoded documents can be stored in a unified manner transcode_ouput Directory. If you do not need to set a prefix, use an empty string.

- (Optional) Outputs

Transcoding output file information. At least one of outputs and snapshots must exist.

- Outputs: Key

Transcoding output file name. The output file will be saved in the outputbucket set in the pipeline. be careful: if the output file name already exists in the storage, the original file will be overwritten by the transcoding output file. If the container in the template is ts , If the segmentduration is not empty, the final output file name will be increased based on the provided output file name uuid00000.ts and `.m3u8` . So if the supplied output file name already contains a suffix `.ts` , the end result will be `OutputKeyPrefixKey.tsuuid00000.ts` , `uuid` represents the unique identifier of this transcoding, and all `ts` files will be carried, similar to `471c1c6e620c45debe9e3b3388749017`

- Outputs: PresetId

The transcoding template ID of this output application.

- (Optional, TS output only) Outputs: SegmentDuration

HLS slice duration. Optional values: 1 - 60. The unit is seconds. Note that segmentduration is only valid for containers `ts` effective. For other containers, this item is automatically ignored.

- (Optional) Outputs: Watermarks

Video watermarking. If this option is configured, the video coding method in the transcoding template cannot be "unchanged", otherwise the task cannot be created. Currently, only a single watermark is supported, for example:

```
"Watermarks": [{
    "InputKey": "abc.png",
}]
```

The watermark location information is configured in the transcoding template. The default is the upper right corner. The picture format only supports JPG and PNG. "Inputkey" refers to the image file key under the input bucket (configured in the pipeline), which must exist.

- (Optional, HLS encryption) Outputs: Encryption

HLS slice AES-128 encrypted information. This option only works when HLS slices.

```
"Encryption": {
    "Mode": "AES128",
    "Key": "string",
    "KeyMd5": "string",
    "InitializationVector": "string"
}
```

"Mode" option: currently only "aes128" is supported. "Key" option: a string in JSON format, including the secret key of aes128 encrypted video (must be 16 bytes) and the access URL of the key. It must be Base64 encoded. For example: '{"URL":“ http://ss.bscstorage.com/file.key "," key ":" 1111111122222222222 "} ', Base64 encoding is followed by:' eyjvcmwioiiaiahr0cdovl3nzlmjzy3n0b3jhz2uuy29tl2zpbgua2v5iiwgiktlesi6icixmtex mtexmtexmtiymjiymjiyiin0 = '" keymd5 "option: MD5 value of key not Base64 encoded, which must be Base64 encoded." Initializationvector "option: optional configuration. The initialization vector is 16 bytes long and must be Base64 encoded.

- (Optional, video clipping) Outputs: Composition

Video clipping configuration, currently only supports single clipping.

```
"Composition": [{
  "TimeSpan": {
    "StartTime": "string",
    "Duration": "string"
  }
}]
```

"Starttime" option: the start time of clipping, in milliseconds. "Duration" option: the duration of clipping, in milliseconds.

- （Optional）Snapshots

Screenshot output information. At least one of Outputs and Snapshots must exist

- Snapshots: Format

Screenshot file format. Optional value：png, jpg

- Snapshots: Key

Output screenshot file name. Note that the extension is automatically added according to the format.

- Snapshots: Time

Screenshot start time. The format is s.S (s is seconds, s is floating-point milliseconds).

- （Optional）Snapshots: Interval

Screenshot interval. The format is s.S (s is seconds, s is floating-point milliseconds). It works only when number > 1. If the screenshot interval is set, the picture index will be automatically added to the output file name, such as

snapshot0000.png - snapshop0010.png

。

- （Optional）Snapshots: Number

Maximum number of screenshots. The default is 1.

- （Optional）Snapshots: Resolution

Screenshot resolution. The format is [width] x [height]. For example: 1920x1080. The default is the original video resolution.

- （Optional）Snapshots: AspectRatio

Screenshot aspect ratio. The optional values are ["1:1", "4:3", "3:2", "16:9"]. The default is the original video aspect ratio.

- （Optional）Playlists

Adaptive HLS transcoding output information.

- Playlists: Format

Adaptive transcoding format. Optional values: hlsv3, hlsv4. For each sub HLS transcoding in the adaptive HLS playlist

- HLSv3 Transcoding generates TS slice files (video00000.ts, video00001. Ts,...) and a m3u8 playlist.
- HLSv4 Transcoding generates a TS file and a m3u8 playlist.
- Playlists: Name

The file name of the adaptive HLS playlist. Note that the system will automatically add `.m3u8` Extension.

Playlists: OutputKeys

The output file name included in the adaptive HLS playlist. Each contained output must be HLS output (Container is `ts`， and SegmendDration is provided），And the SegmentDuration must be the same.

- PipelineId

Pipeline ID used by this transcoding task

Return

```
Status: 201 Created
Content-Type: application/json
Content-Length: number of characters in the response
Date: Mon, 14 Jan 2013 06:01:47 GMT

{
   "Job":{
      "Id":"任务 ID",
      "Inputs":[{
         "Key":"转码源文件名"
      }],
      "OutputKeyPrefix":"转码后的输出文件名前缀",
      "Outputs":[{
         "Key":"转码后的输出文件名",
         "PresetId":"转码模板 ID",
         "SegmentDuration":"[1,60]",
      },
      {...}],
      "Snapshots":[{
         "Key":"转码后的输出文件名",
         "Format":"jpg|png",
         "Time":"截图开始时间点",
         "Interval":"截图间隔时间",
         "Number":"截图数量",
      },
      {...}]
      "Playlists":[{
         "Format":"HLSv3|HLSv4",
         "Name":"播放列表文件名",
         "OutputKeys":[
            "包含在输出文件列表中的输出文件名",
            ...
         ],
      },
      {...}],
      "PipelineId":"本转码任务使用的管道 ID",
      "Status":"Submitted|Progressing|Complete|Canceled|Error",
      "Timing":{
          "SubmitTimeMillis":"任务创建的时间。单位：epoch milliseconds",
          "FinishTimeMillis":"任务完成的时间。单位：epoch milliseconds"
      }
   }
}
```

example

```
{
   "Inputs":[{
      "Key":"input/test.ts"
   }],
   "OutputKeyPrefix":"",
   "Outputs":[{
      "Key":"test.mp4",
      "PresetId":"123",
   }],
   "PipelineId":"1000000000000000001"
}
```

Create tasks through SDK

python

```python
import boto3
from botocore.client import Config


config = Config(signature_version='s3v4')
cli = boto3.client('elastictranscoder',
                   config=config,
                   region_name='us-west-2',
                   endpoint_url='http://transcoder-ss.bscstorage.com',
                   aws_access_key_id="accesskey",
                   aws_secret_access_key="secretkey")

inputs = [{
   'Key': "sample.mp4",
}]

outputs = [{
   'Key': "sample.m3u8",
   # 转码模板id，需要预先配置
   'PresetId': "1000",
   'SegmentDuration': "10",
}]

print cli.create_job(
   # 管道id，请预先创建
   PipelineId='pipelineid',
   Inputs=inputs,
   Outputs=outputs,
)
```

java

```java
public static void create_job() {
   BasicAWSCredentials awsCreds = new BasicAWSCredentials("accessKey", "secret

   ClientConfiguration clientconfiguration = new ClientConfiguration();
   clientconfiguration.setSocketTimeout(60 * 60 * 1000); // in milliseconds
   clientconfiguration.setConnectionTimeout(60 * 60 * 1000); // in millisecond

   AmazonElasticTranscoder client = new AmazonElasticTranscoderClient(awsCreds
   client.setEndpoint("http://transcoder-ss.bscstorage.com");

   JobInput input = new JobInput().withKey("sample.mp4");

   List<JobInput> inputs = Arrays.asList(input);

   CreateJobOutput out1 = new CreateJobOutput().withKey("output.mp4").withPres

   List<CreateJobOutput> outputs = Arrays.asList(out1);

   CreateJobRequest createJobRequest = new CreateJobRequest().withPipelineId("
            .withInputs(inputs).withOutputs(outputs);

   Job j = client.createJob(createJobRequest).getJob();
   System.out.printf("task id: %s\n", j.getId());
}
```

Get task

Describe

Send GET request to `/2012-09-25/jobs/jobId` to get detailed task information.

Request

```
GET /2012-09-25/jobs/jobId HTTP/1.1
Content-Type: charset=UTF-8
Accept: */*
Host: transcoder-ss.bscstorage.com
x-amz-date: 20170726T174952Z
Authorization: AWS4-HMAC-SHA256
               Credential=AccessKeyID/request-date/Transcoder endpoint/elastic
               SignedHeaders=host;x-amz-date;x-amz-target,
               Signature=calculated-signature
```

return

```
Status: 200 OK
Content-Type: application/json
Content-Length: number of characters in the response
Date: Mon, 14 Jan 2013 06:01:47 GMT

{
    "Job":{
        "Id":"任务 ID",
        "Inputs":[{
            "Key":"转码源文件名"
        }],
        "OutputKeyPrefix":"转码后的输出文件名前缀",
        "Outputs":[{
            "Key":"转码后的输出文件名",
            "PresetId":"转码模板 ID",
            "SegmentDuration":"[1,60]",
        },
        {...}],
        "Snapshots":[{
            "Key":"转码后的输出文件名",
            "Format":"jpg|png",
            "Time":"截图开始时间点",
            "Interval":"截图间隔时间",
            "Number":"截图数量",
        },
        {...}]
        "Playlists":[{
            "Format":"HLSv3|HLSv4",
            "Name":"播放列表文件名",
            "OutputKeys":[
               "包含在输出文件列表中的输出文件名",
               ...
            ],
        },
        {...}],
        "PipelineId":"本转码任务使用的管道 ID",
        "Status":"Submitted|Progressing|Complete|Canceled|Error",
        "Timing":{
            "SubmitTimeMillis":"任务创建的时间。单位：epoch milliseconds",
            "FinishTimeMillis":"任务完成的时间。单位：epoch milliseconds"
        }
    }
}
```

Lists the tasks in the pipeline

Description

Send GET request to `/2012-09-25/jobsByPipeline/pipelineId` list the tasks assigned in the specified pipeline.

Request

```
GET /2012-09-25/jobsByPipeline/pipelineId?PageToken=12345678 HTTP/1.1
Content-Type: charset=UTF-8
Accept: */*
Host: transcoder-ss.bscstorage.com
x-amz-date: 20170726T174952Z
Authorization: AWS4-HMAC-SHA256
               Credential=AccessKeyID/request-date/Transcoder endpoint/elastic
               SignedHeaders=host;x-amz-date;x-amz-target,
               Signature=calculated-signature
```

Request parameters

- PageToken

If nextpagetoken appears in the result, it indicates that the result column is not complete. You need to use the nextpagetoken in the previous result as the pagetoken in the next request to continue the request result.

Return

```
Status: 200 OK
Content-Type: application/json
Content-Length: number of characters in the response
Date: Mon, 14 Jan 2013 06:01:47 GMT

{
   "Jobs":[{
      "Id":"任务 ID",
      "Inputs":[{
          "Key":"转码源文件名"
      }],
      "OutputKeyPrefix":"转码后的输出文件名前缀",
      "Outputs":[{
         "Key":"转码后的输出文件名",
         "PresetId":"转码模板 ID",
         "SegmentDuration":"[1,60]",
      },
      {...}],
      "Snapshots":[{
         "Key":"转码后的输出文件名",
         "Format":"jpg|png",
         "Time":"截图开始时间点",
         "Interval":"截图间隔时间",
         "Number":"截图数量",
      },
      {...}]
      "Playlists":[{
         "Format":"HLSv3|HLSv4",
         "Name":"播放列表文件名",
         "OutputKeys":[
            "包含在输出文件列表中的输出文件名",
            ...
         ],
      },
      {...}],
      "PipelineId":"本转码任务使用的管道 ID",
      "Status":"Submitted|Progressing|Complete|Canceled|Error",
      "Timing":{
         "SubmitTimeMillis":"任务创建的时间。单位：epoch milliseconds",
         "FinishTimeMillis":"任务完成的时间。单位：epoch milliseconds"
      }
   },
   {...}],
  "NextPageToken": "1234567890"
}
```

Cancel task

Description

Send DELETE request to `/2012-09-25/jobs/jobId` to cancel the created task.
Note: tasks that have been started cannot be cancelled.

Request

```
DELETE /2012-09-25/jobs/jobId HTTP/1.1
Content-Type: charset=UTF-8
Accept: */*
Host: transcoder-ss.bscstorage.com
x-amz-date: 20170726T174952Z
Authorization: AWS4-HMAC-SHA256
               Credential=AccessKeyID/request-date/Transcoder endpoint/elastic
               SignedHeaders=host;x-amz-date;x-amz-target,
               Signature=calculated-signature
```

Return

```
Status: 202 Accepted
Content-Type: application/json
Content-Length: number of characters in the response
Date: Mon, 14 Jan 2013 06:01:47 GMT

{
   "Success":"true"
}
```

Obtain audio and video meta information

Describe

Send GET request to `/metadata/<bucket>/<key>` get meta information.

Request

```
GET /metadata/test_bucket/test.mp4 HTTP/1.1
Content-Type: charset=UTF-8
Accept: */*
Host: transcoder-ss.bscstorage.com
x-amz-content-sha256: UNSIGNED-PAYLOAD
x-amz-date: 20170925T093131Z
Authorization: AWS4-HMAC-SHA256
               Credential=acc_yanhui/20170925/us-east-1/elastictranscoder/aws4
               SignedHeaders=host;x-amz-content-sha256;x-amz-date,
               Signature=e9147d9d1461d7ca92c0805735490be7fd7d9d92ddee216ee4730
```

Request parameters

- bucket

bucket name

- key

Key of audio and video file to obtain meta information

How to use SDK interface

python

```python
import boto3
from botocore.client import Config
import json


cli = boto3.client(
    's3',
    use_ssl=False,
    aws_access_key_id='accesskey',
    aws_secret_access_key='secretkey',
    endpoint_url='http://transcoder-ss.bscstorage.com/metadata',
    config=Config(s3={'addressing_style': 'path'})
)

res = cli.get_object(Bucket='your bucket', Key="sample.mp4")
res_json = json.loads(res['Body'].read())
print repr(res_json)
```

java

```java
public static void get_meta() {
    BasicAWSCredentials awsCreds = new BasicAWSCredentials("accesskey", "secre
    ClientConfiguration clientconfiguration = new ClientConfiguration();
    clientconfiguration.setSocketTimeout(60 * 60 * 1000); // in milliseconds
    clientconfiguration.setConnectionTimeout(60 * 60 * 1000); // in millisecon

    AmazonS3 client = new AmazonS3Client(awsCreds, clientconfiguration);
    client.setEndpoint("http://transcoder-ss.bscstorage.com/metadata");

    S3ClientOptions opt = S3ClientOptions.builder().setPathStyleAccess(true).b
    client.setS3ClientOptions(opt);

    S3Object s3Object = client.getObject("your bucket", "sample.mp4");
    byte[] buf = new byte[10240];
    try {
        s3Object.getObjectContent().read(buf, 0, 10240);
        System.out.println(new String(buf, "UTF-8"));
    } catch (IOException e) {
        System.out.println("errr");
    }
}
```

Return

```
Status:200

Content-length: '2253'
x-amz-s2-requester: test_bucket
server: openresty/1.11.2.4
connection: keep-alive
x-amz-request-id: 00361611-1709-2517-4651-00163e0630f7
date: Mon, 25 Sep 2017 09:46:51 GMT
content-type: application/json

{
    "audio_streams":    音频信息
        [
            {
                "sample_fmt": "fltp",   采样格式
                "codec_tag": "0x6134706d",  编码器标签
                "codec_type": "audio",  编码器类型
                "channels": 2,  音频数
                "bit_rate": "96001",    码率
                "codec_name": "aac",    编码器名
                "duration": "716.885011",   文件总时长
                "nb_frames": "30876",   帧数
                "codec_time_base": "1/44100",   编码器每帧时长
                "index": 1, 流索引号
                "start_pts": 0, 时间戳
                "profile": "LC", 编码的profile
                "tags": 标签信息
                    {
                        "handler_name": "SoundHandler", 处理器名字
                        "language": "und"   语言
                    },
                "r_frame_rate": "0/0",  真实基础帧率
                "start_time": "0.000000",   首帧时间
                "time_base": "1/44100", 每帧时长
                "codec_tag_string": "mp4a", 编码器标签名
                "duration_ts": 31614629, 单位为time_base的时长
                "codec_long_name": "AAC (Advanced Audio Coding)",   编码器名全称
                "bits_per_sample": 0,   采样码率
                "avg_frame_rate": "0/0",    平均帧率
                "channel_layout": "stereo", 声道
                "max_bit_rate": "96001", 最大码率
                "sample_rate": "44100"  采样率
            }
        ],

    "video_streams":    视频信息
        [
            {
                "profile": "High",  编码的profile
                "sample_aspect_ratio": "0:1",
                "codec_tag": "0x31637661",  编码器标签
                "refs": 1,
                "codec_type": "video",  编码器类型
                "coded_height": 720,    图像高度
                "bit_rate": "1500443",  码率
                "codec_name": "h264",   编码器名
                "duration": "716.800000",   时长
                "is_avc": "true",
                "nb_frames": "17920",   帧数
                "codec_time_base": "1/50",  编码器每帧时长
                "index": 0, 流索引号
                "start_pts": 0,
                "width": 1280,  帧宽度
                "coded_width": 1280,    图像宽度
                "pix_fmt": "yuv420p",   像素个数
```

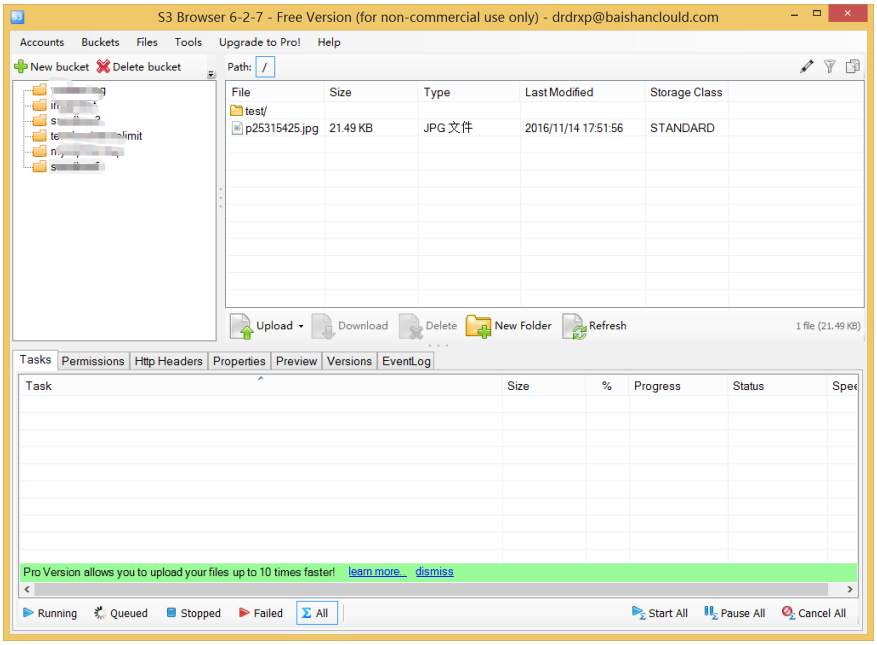
```
                "chroma_location": "left",
                "tags":
                    {
                        "handler_name": "VideoHandler", 处理器名字
                        "language": "und"   语言
                    },
                "r_frame_rate": "25/1", 真实基础帧率
                "start_time": "0.000000",   首帧时间
                "time_base": "1/12800", 每帧时长
                "codec_tag_string": "avc1", 编码器标签名
                "duration_ts": 9175040, 单位为time_base的时长
                "codec_long_name": "H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10"
                "display_aspect_ratio": "0:1",
                "height": 720,  帧高度
                "avg_frame_rate": "25/1",   平均帧率
                "level": 40,    级别
                "bits_per_raw_sample": "8",
                "has_b_frames": 2,  记录缓存帧大小
                "nal_length_size": "4"
            }
        ],

    "format":
        {
            "tags": 标签信息
                {
                    "major_brand": "isom",
                    "minor_version": "512",
                    "compatible_brands": "isomiso2avc1mp41letv",
                    "encoder": "Lavf56.15.102"
                },
            "nb_streams": 2,    流的个数
            "start_time": "0.000000",   首帧时间
            "format_long_name": "QuickTime / MOV",  格式名全称
            "format_name": "mov,mp4,m4a,3gp,3g2,mj2",   格式名
            "bit_rate": "1607943",  码率
            "nb_programs": 0,
            "duration": "716.932000",   时长
            "size": "144098269" 文件大小
        },

    "other_streams": []
}
```

Storage Management Tools

Steps of using S3 browser to connect Baishan storage and upload files under Windows

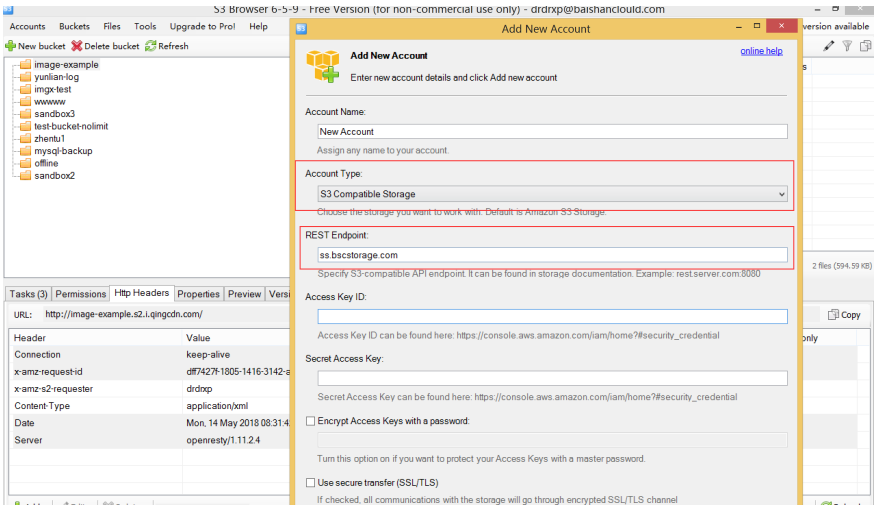
Step 1

Open S3 browser software, and the interface is shown in the following figure:



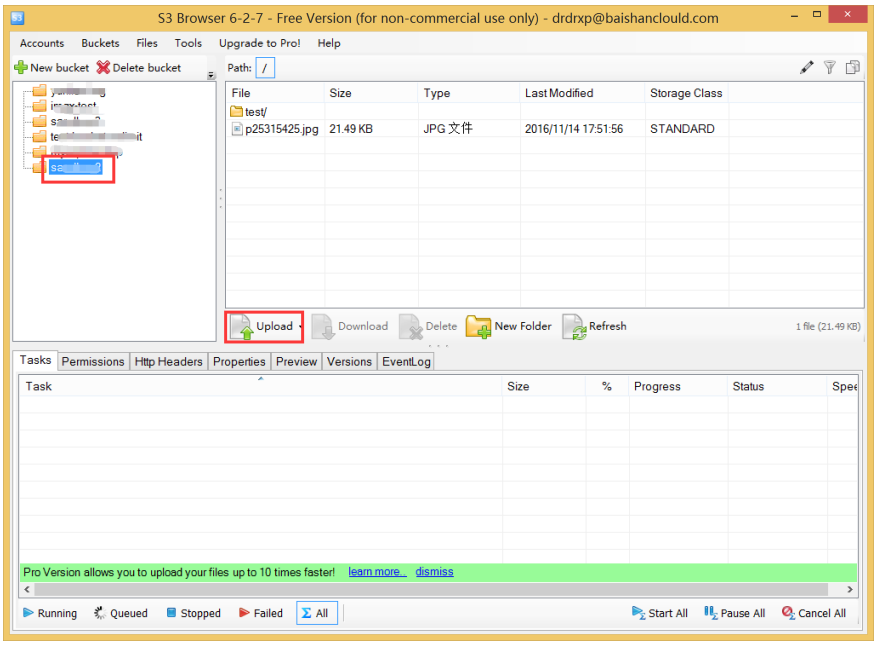
Step 2

Click add new account under accounts to open the following interface. Note that select S3 compatible storage for storage type and SS for rest endpoint bscstorage. com。 Enter the account, accessKey and secret key, and then click to add a new account.



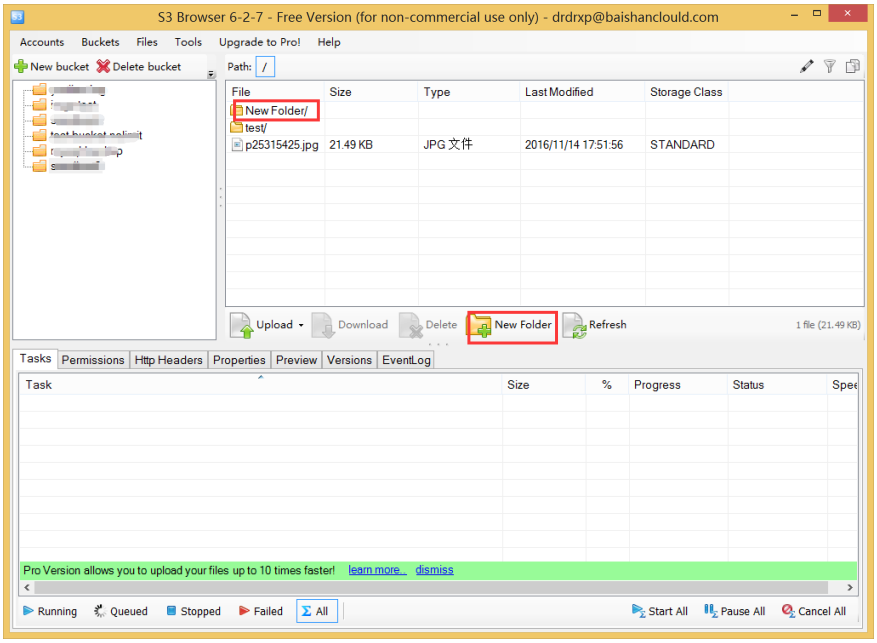
Step 3

After adding a new account, you can see the contents of storage. Select a bucket on the left and click upload to upload files to the bucket.



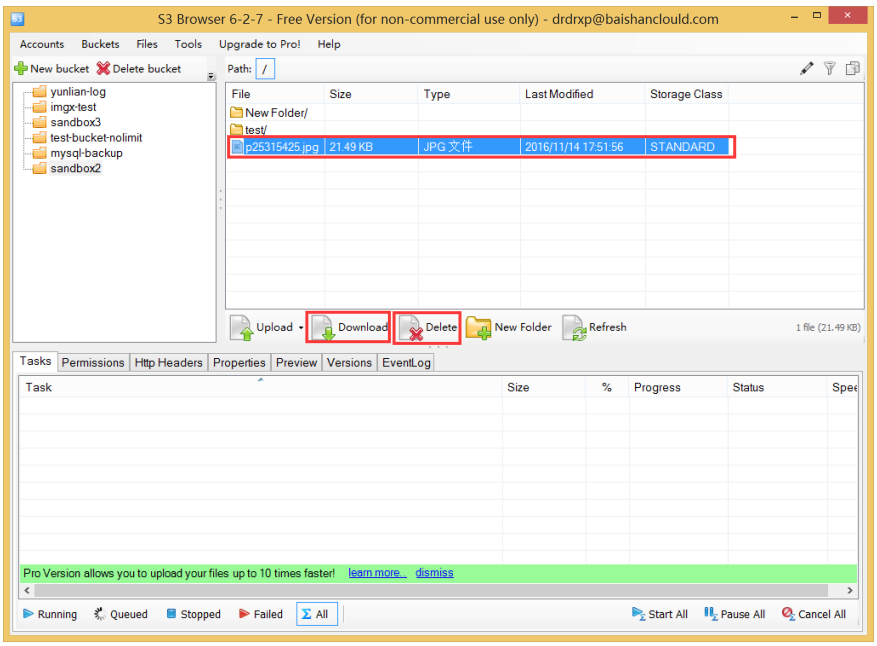
Step 4

Select a bucket on the left and click new folder to create a new folder in this bucket.



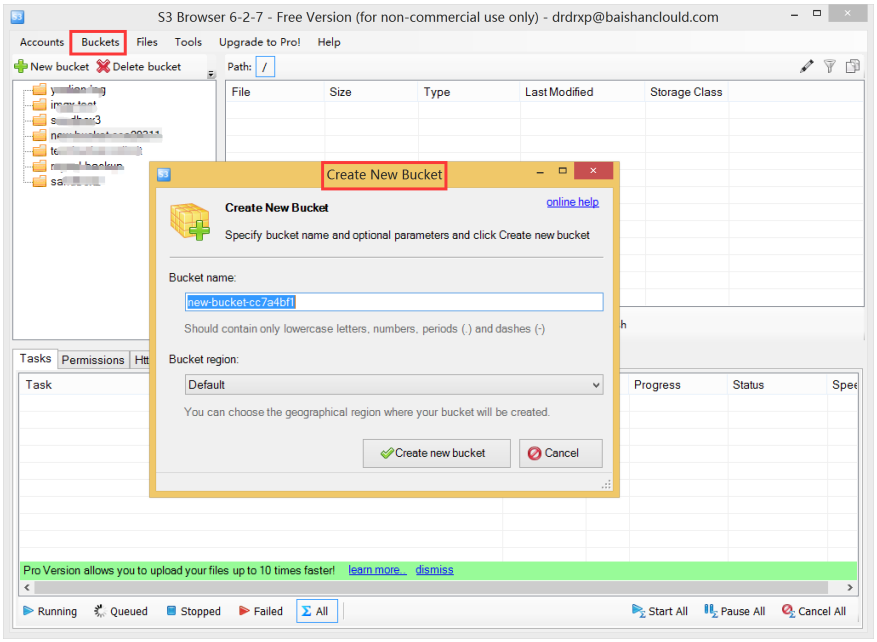
Step 5

Select a file and click download and delete to download and delete the file.



Step 6

You can also click the buckets menu to add a bucket and other operations. So far, you have learned all the processes of docking Baishan storage and uploading files with S3 browser.



S3 browser download address:s3-browser

s3-browser

Console User Guide

Baishan Cloud Storage Console User Guide

1. Overview

Cloud Storage Console is a management platform for customers to use cloud storage online.

The management platform mainly includes the following functional modules:

My Storage:

- Display all user Buckets.
- Configuration management of Bucket.
- Configure the image processing format.
- Manage the objects in the Bucket.

Statistical analysis: Graphically display data information for each time period such as bandwidth, traffic, number of requests, storage space, etc.

Video Processing: Users can create transcoding templates, configure rules for automatic or active transcoding and view statistical analysis data of video processing.

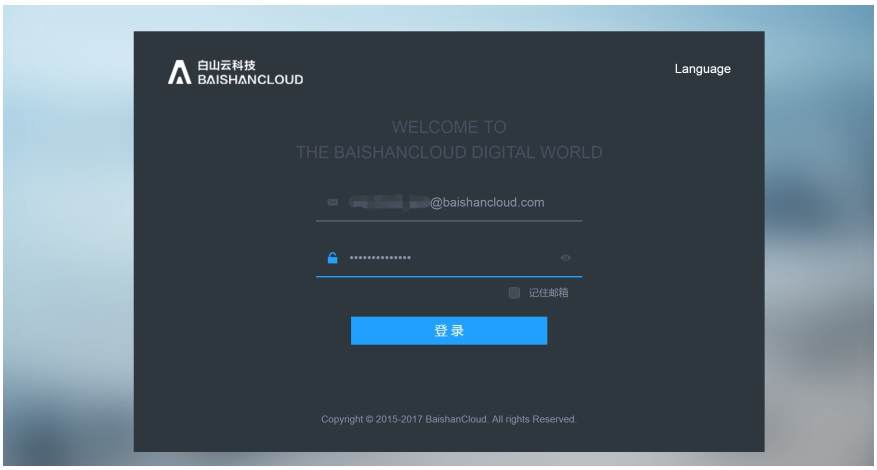
Secret Key Management: Display the Accesskey and SecretKey of the user's current account.

User management: Main account can create sub-accounts and give corresponding permissions to sub-accounts.

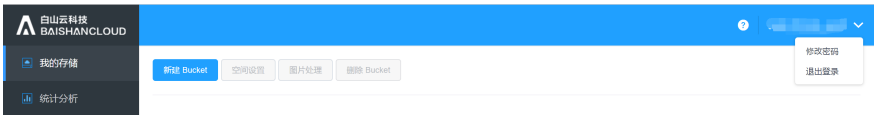
Help documentation: Documentation to help users use the Baishan Cloud Storage console.

1.Functional features

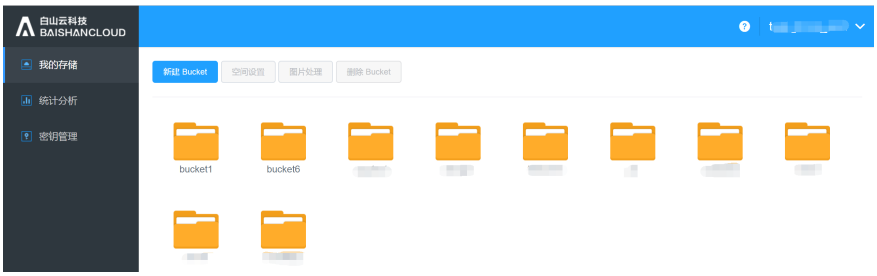
1. User Login Portal URL: http://cwn-ss.bscstorage.com/, enter user name and password to log in.



After logging in, you can choose to log out at the upper right corner.

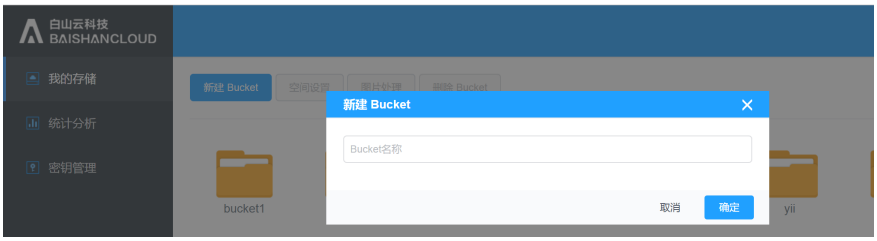


1. My Storage



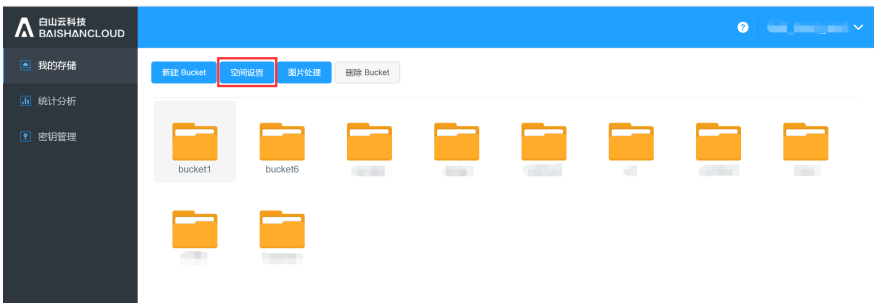
1. New Bucket

User creates a new Bucket and names this Bucket. Bucket naming rules: Unique within the cloud storage account; composed of lowercase letters, numbers or dash symbol '-', length 3~63 digits; cannot start with numbers; cannot start or end with dash symbol '-'.

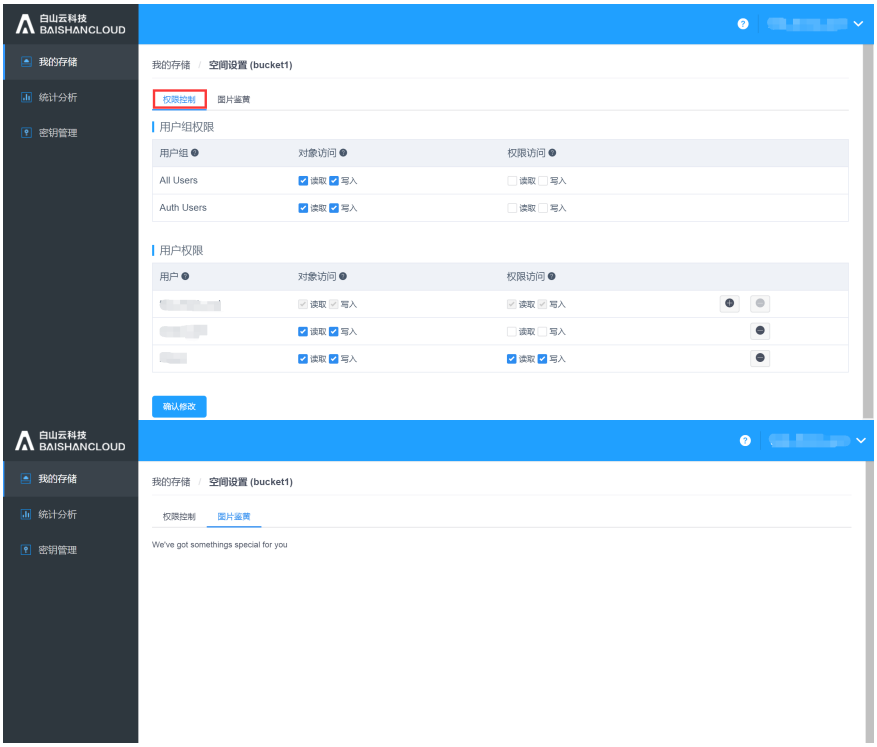


1. Space setting

Select a Bucket and click on "Space Settings".

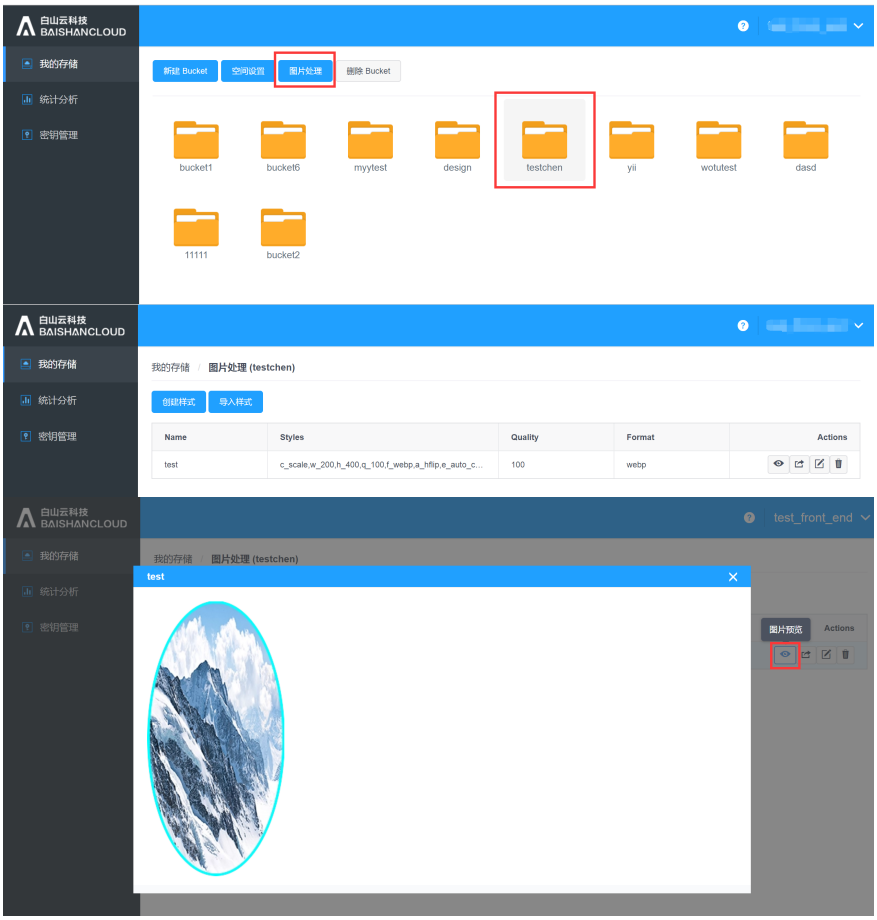


Under Space settings, there are "Permissions" and "Sensitive content filtering", etc. This permission setting applies to the storage space bucket, you can also add permission to certain users.



1. Image processing

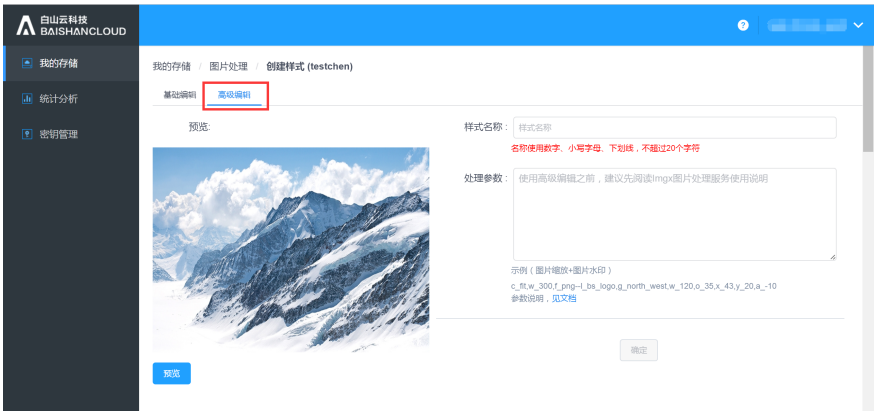
Select a Bucket and click "Image Processing" to enter the image processing management page.



Click "Create Style" to create a new image processing style.

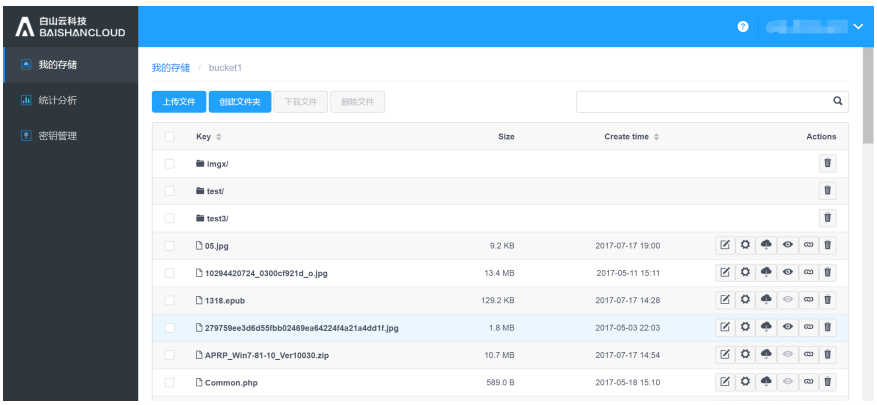
More settings:

Advanced edit:

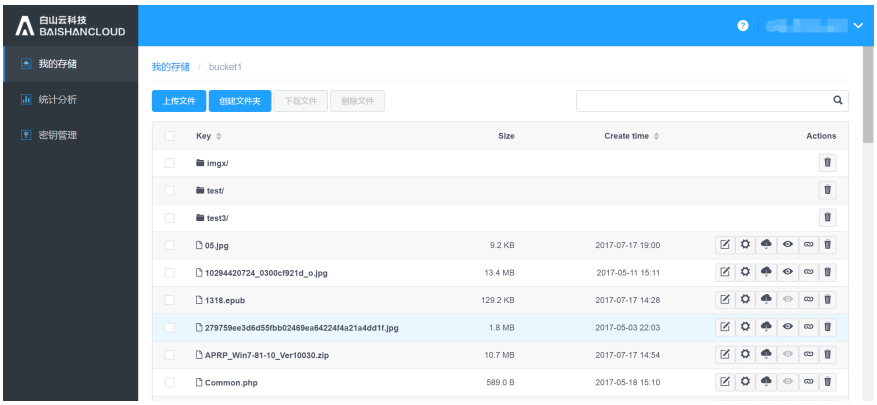


1. File Management

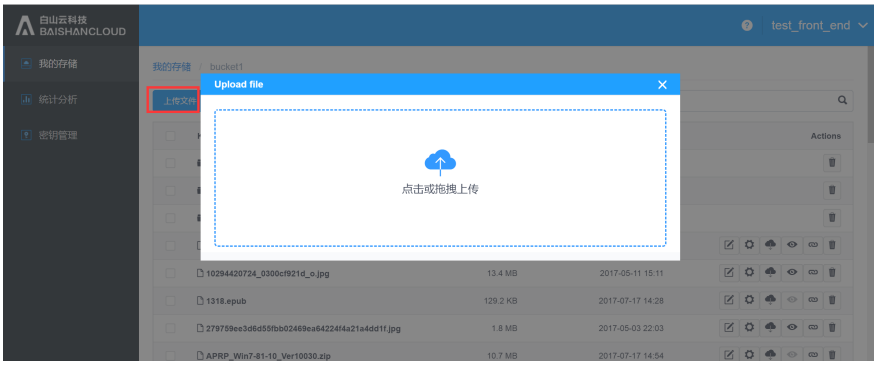
Double-click a Bucket to enter to manage the files or folders in a Bucket.



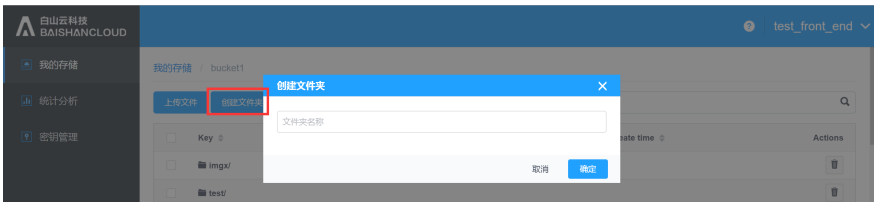
The following shows the file management interface: Users can create/delete folders, upload/delete files, and manage permissions, download, get URL addresses, preview files, and perform other operations here.



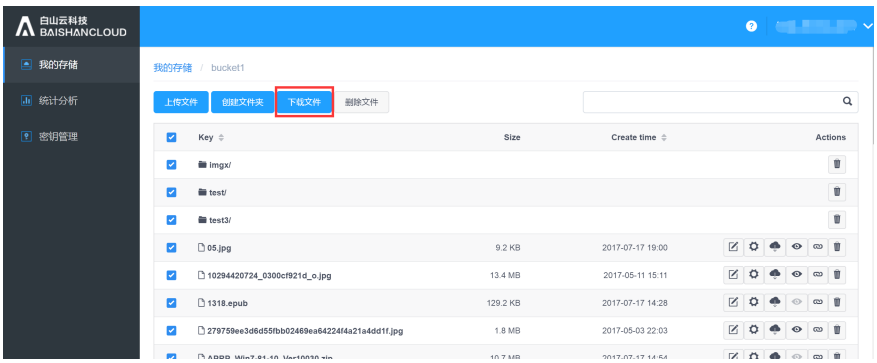
Click “Upload file” to upload local files:



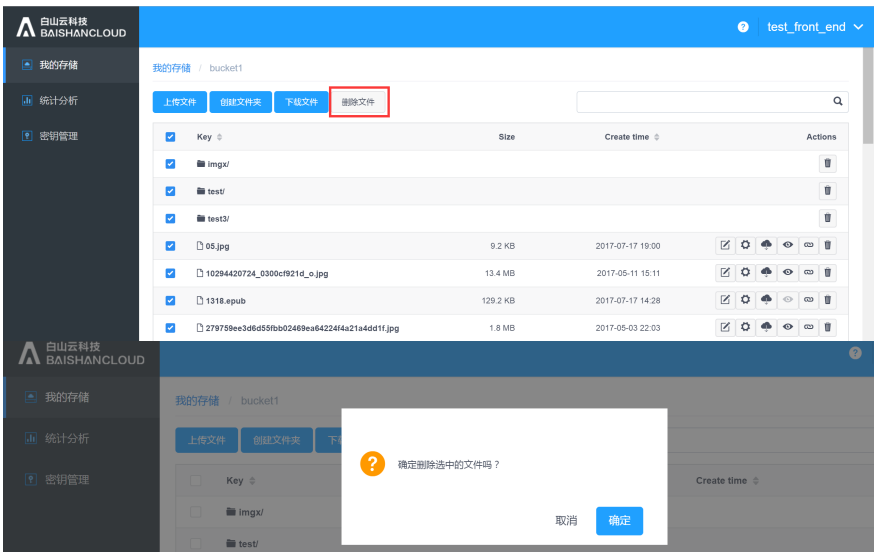
Click “Create folder” to create a new folder in this Bucket:



Click “Download” to download selected files:



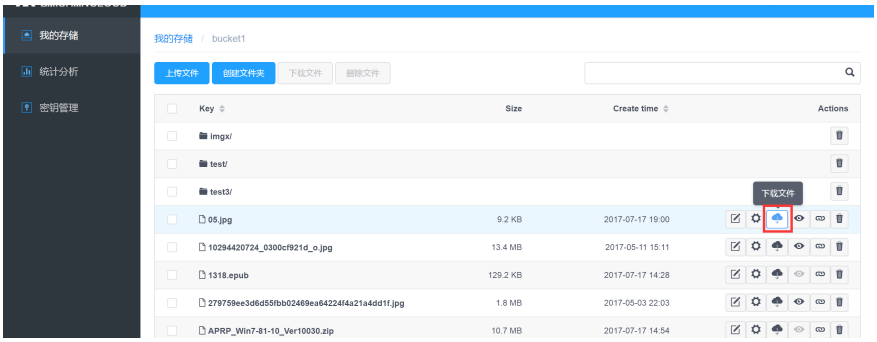
Click “Delete” to delete selected files:



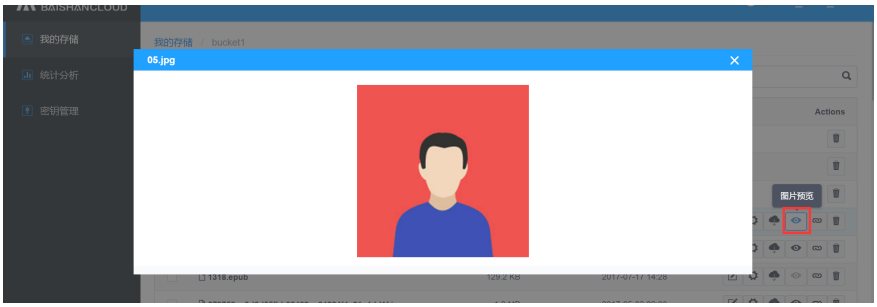
Click “Rename” to rename a file:

Permission setting: Allow users to grant different access/editing permissions.

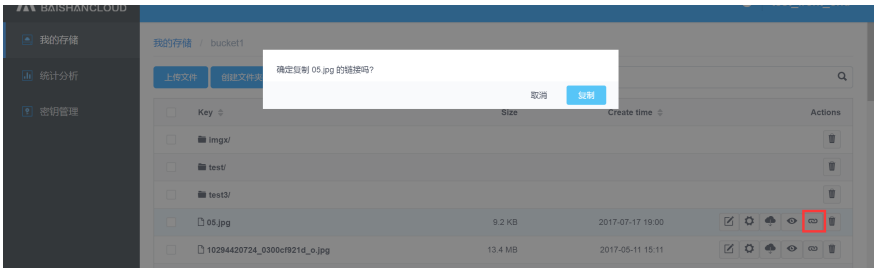
Click “Download” to download files to your local folder:



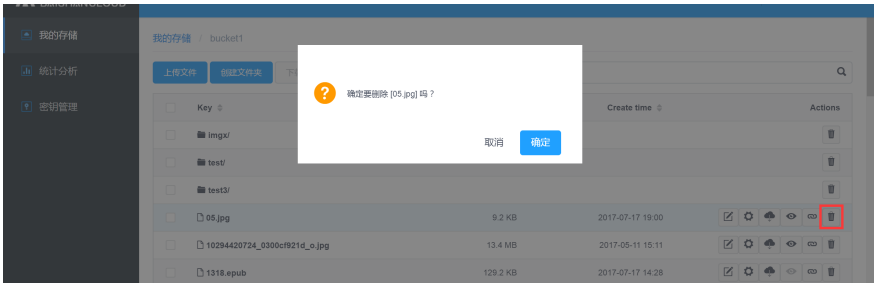
Click “Preview” to preview images:



Click “Get URL” to obtain the URL for this file:

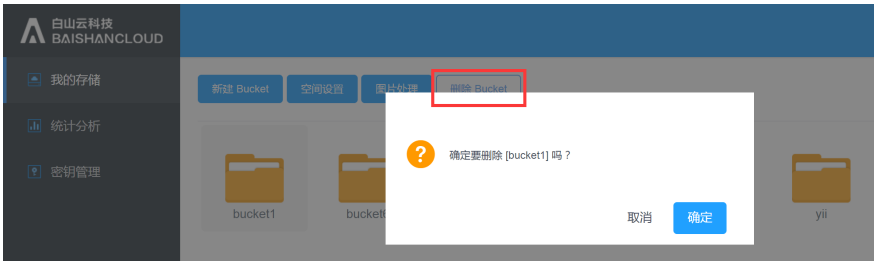


Click “Delete” to delete selected file:



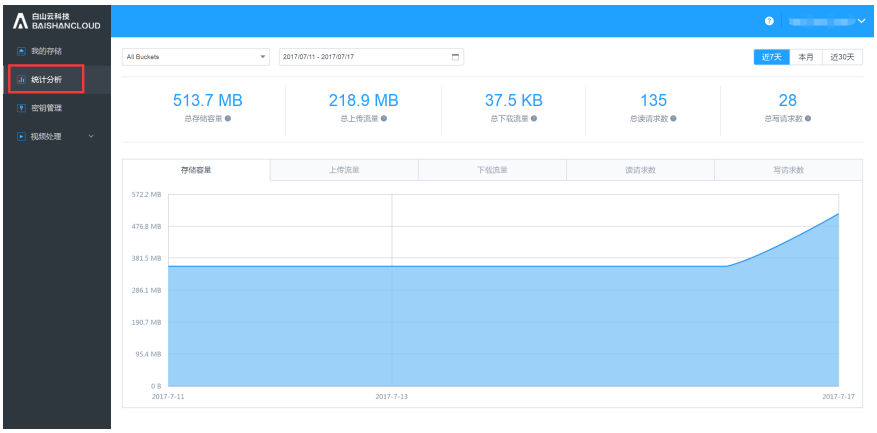
1. Delete Bucket

Users can delete the non-empty Bucket.

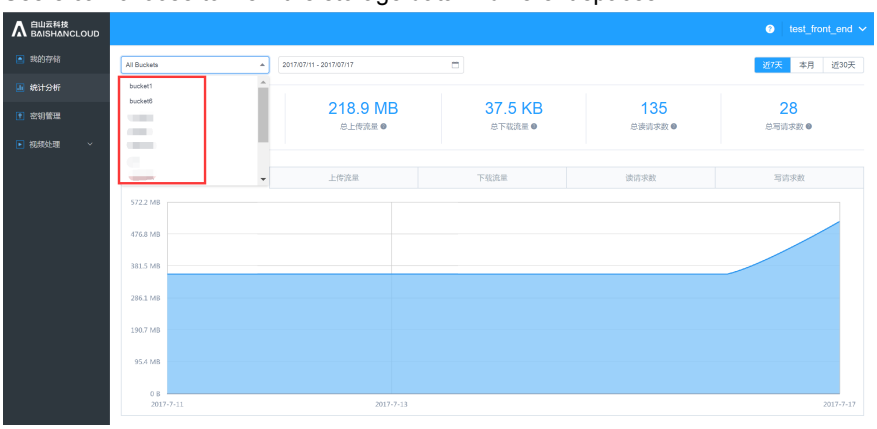


1. Statistical Analysis

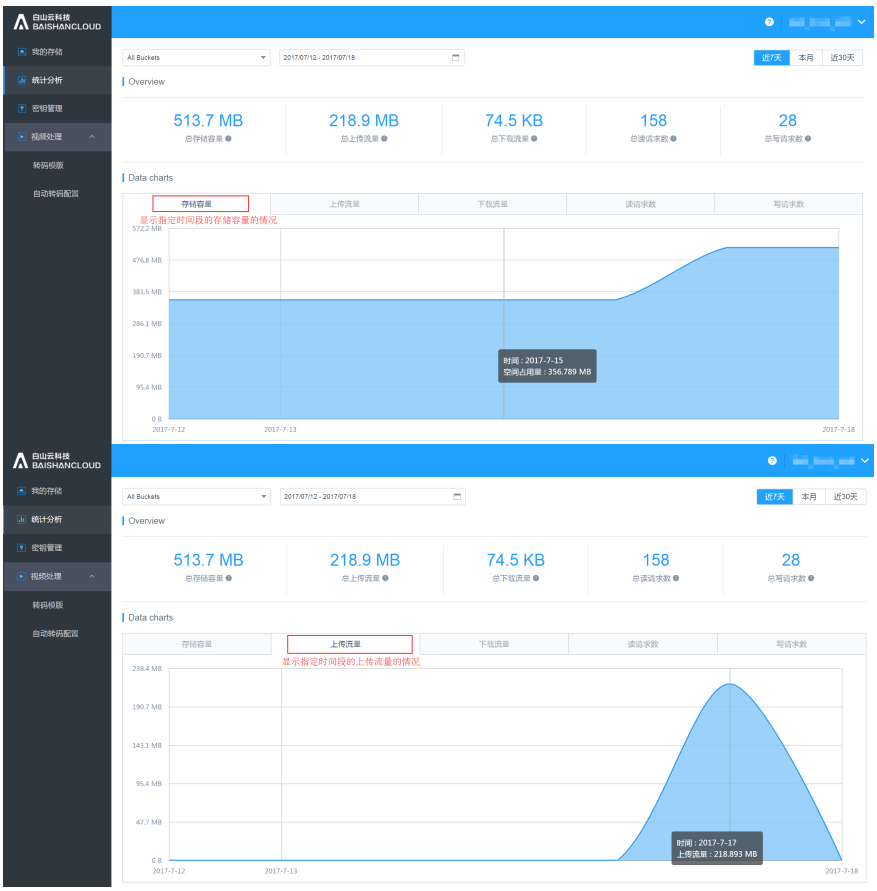
Statistical analysis: It shows the storage situation of users at each time period, including bandwidth, traffic, number of requests, storage space, number of files



Users can choose to view the storage data in different spaces:



Users can view use different filters to lookup the data according to their needs, such as “Storage capacity,” “Uploading traffic,” “Downloading traffic,” “Get request count,” “Post request count,”, examples as following:



1. Video processing

1. Transcoding template

Video templates are transcoding rules that are predefined by users and can be reused. Among them, parameters such as the packaging type, encoding method, resolution, and bit rate after transcoding are specified.

1. Transcoding template management

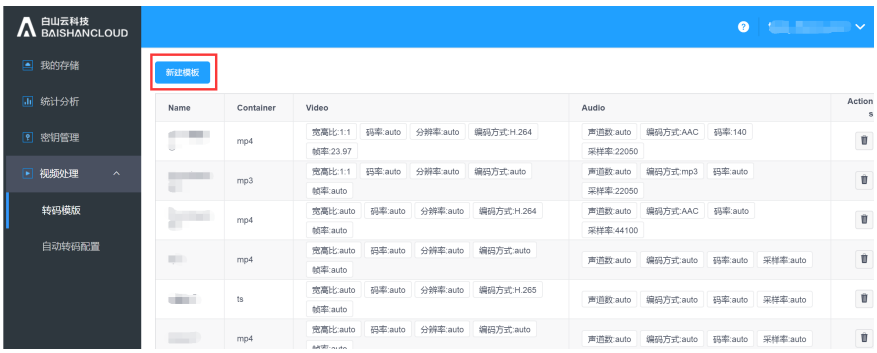
Support creating transcoding templates in the console. Click "Transcoding Template" to enter the transcoding template management page. The basic parameters of video transcoding are listed on the transcoding template

management page. Hover your mouse to the video and audio areas to view detailed parameters.

On this page, to create a new transcoding template, click "New Template".

1. Create a transcoding template

Click "New Template",

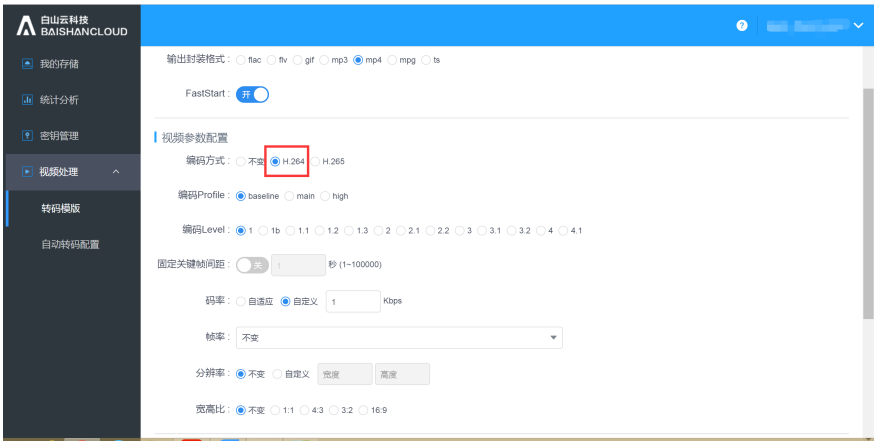


“Create Template” page is as following: Enter "**Template name**", "**Template description**", and check "**Output package format**"; when the output format is mp4, FastStart can be turned on or off. When the output format is not mp4, there is no option for FastStart.

The video configuration parameters are as follows:

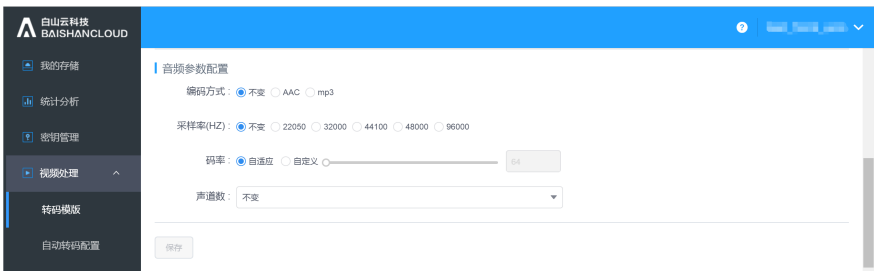
When Encoding is selected "**unaltered**", only "Bit **rate**" and "F**rame rate**" inputs are needed.

When the encoding method is H.264, you need to select "Encoding Profile", "Encoding Level", "Fixed Key Frame Spacing", "Bit Rate", "Frame Rate", "Resolution", and "Aspect Ratio".



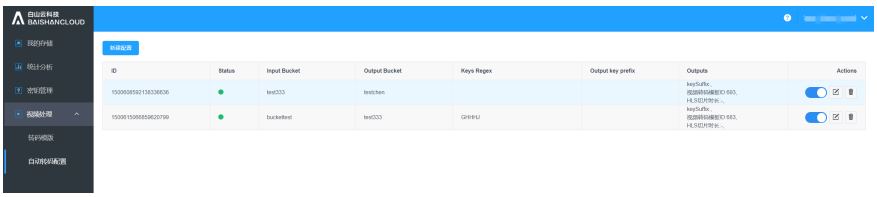
When the encoding method is H.265, you need to select "Fixed Key Frame Spacing", "Code Rate", "Frame Rate", "Resolution", and "Aspect Ratio".

The audio configuration parameters are as follows:



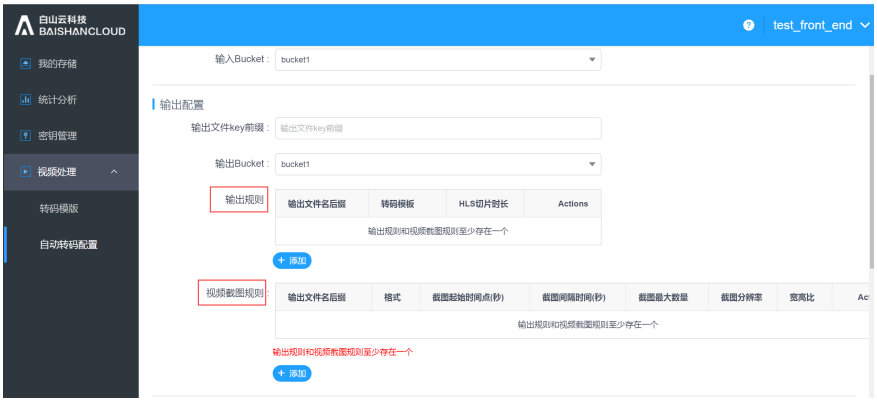
1. Automatic transcoding configuration

The automatic transcoding configuration is suitable for transcoding newly uploaded files in a workflow way. Please note that the newly created automatic transcoding configuration is only valid for newly uploaded files and will not correspond to the files uploaded before the configuration is created. Click "Automatic transcoding configuration" to enter the management page to start.

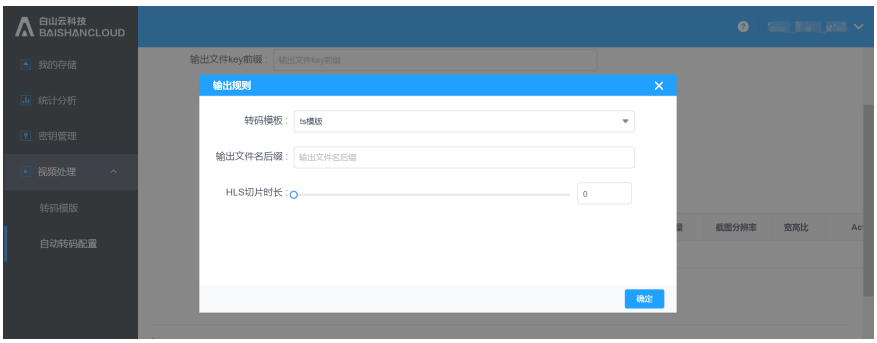


Click "New configuration" to enter the page to create a new configuration. First, create " Path rules that allow transcoding " and "Input Bucket". For "path rules that allow transcoding", you can choose to enter according to the file extension name or regular expression; when configuring the path according to the file extension, you can enter multiple extensions (use "| " between multiple extensions); Only one path can be entered.

Output configuration: Enter the output file name prefix (it can be empty) and the output bucket. At least one output rule or video screenshot rule is required, and multiple sets of output rules and video screenshot rules can be configured.



The output rule configuration page is as follows: When the transcoding template chooses to output ts format, users need to select the HLS slice duration; for other output format only needs to fill in the transcoding template and the output file name suffix.



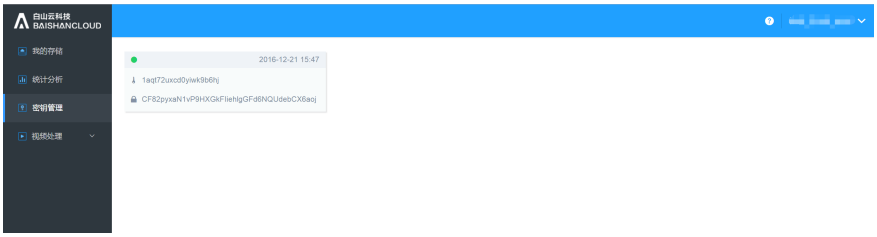
The configuration page for video screenshots is as follows: Require to enter "Output file name suffix", "Screenshot start time", "Screenshot interval", "Maximum number of screenshots", "Resolution" and "Aspect ratio".

Please note at least one of the “Output Rules” and “Screenshot Rules” option is required to be configured. When the transcode module has ts file for output, users can choose whether to enable HLS self-adapt transcoding, if this service is enabled, slicing format and MasterPlayList file suffix are required.

In “More configuration” you can configure options such as “Delete origin files after transcoding,” “Keep origin file path after transcoding,” “Recall URL after failed to transcode,” “Recall URL after successfully transcoded.” Permission settings: Set permission for the output transcoding files

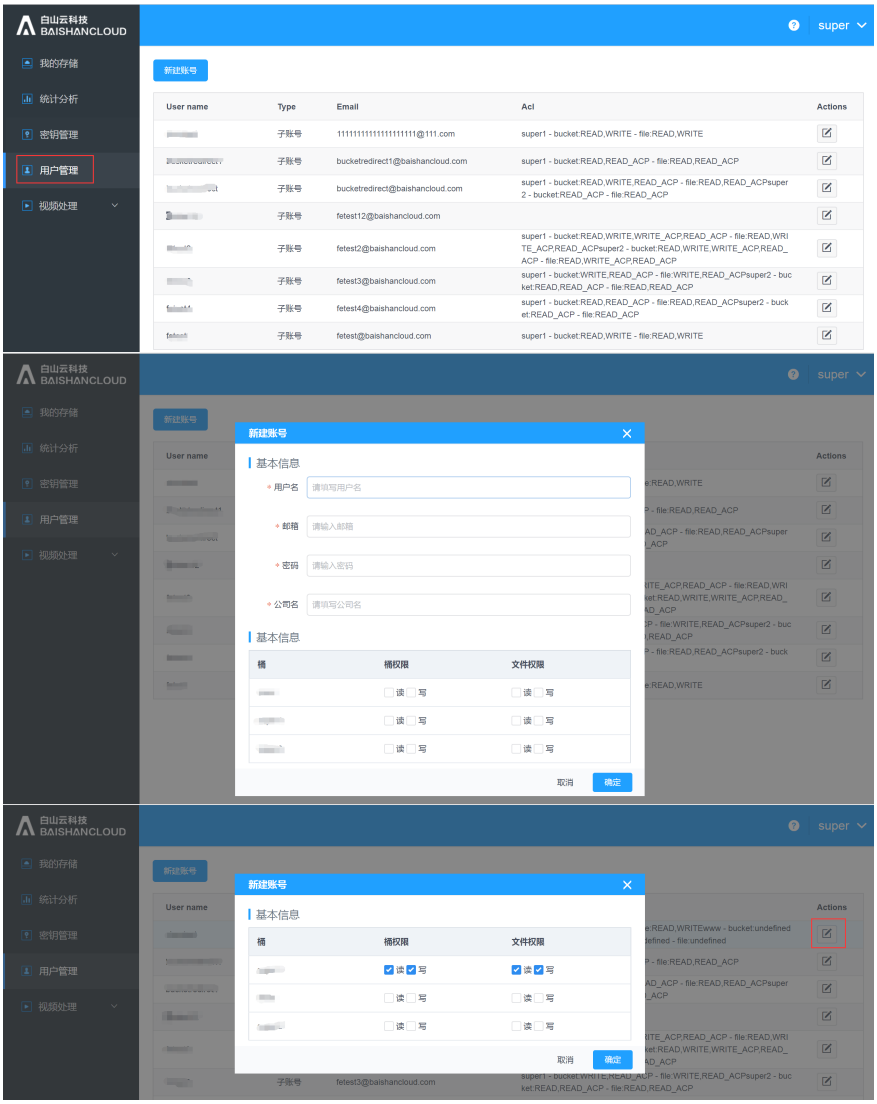
1. Key management

To check the key for this account and the creation time:



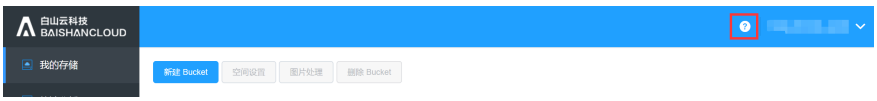
1. User Management

If you are using the primary account, you can create sub-accounts and assign the corresponding permissions to each sub-account.

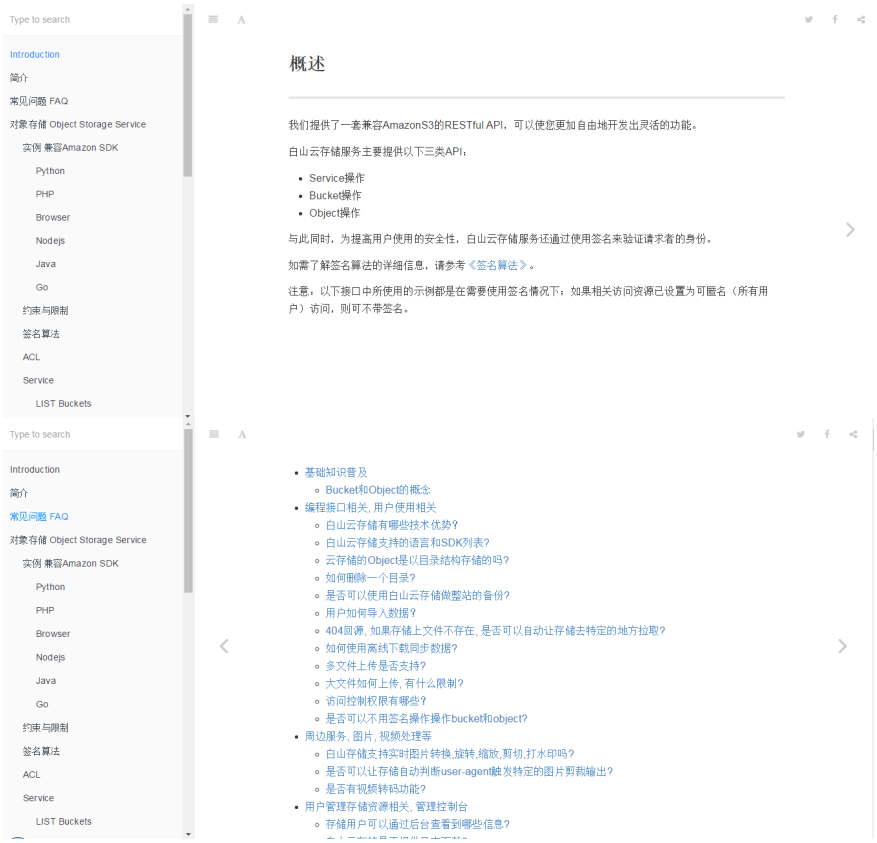


1. Help Documentation

Help documentation provide documents for users to quickly lookup FAQ, details about each function, user cases regarding SDK, call method interface on creating a bucket, acquiring bucket list, upload and download files, image processing sub-service introduction, and description on image uploading tools and method, etc.



To review the details of help documentation description and FAQ, please see following:



For more example of SDK user case, please see following: