Alibaba Cloud
Object Storage Service

Best Practices

Issue: 20190919

Legal disclaimer

Generic conventions

Table -1: Style conventions

Style	Description	Example
⊖	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	⊖ Danger: Resetting will result in the loss of user configuration data.
⚠	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	⚠ Warning: Restarting will cause business interruption. About 10 minutes are required to restore business.
📋	This indicates warning informatio n, supplementary instructions, and other content that the user must understand.	ⓘ Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	📋 Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus , page names, and other UI elements.	Click OK.
`Courier font`	It is used for commands.	Run the `cd / d C :/ windows` command to enter the Windows system folder.
Italics	It is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *[-all|-t]*

Style	Description	Example
{} or {a|b}	**It indicates that it is a required value, and only one item can be selected.**	`swich` *{stand | slave}*

Contents

1 Migrate data to OSS

1.1 Migrate data between buckets in OSS

This topic describes how to use Alibaba Cloud Data Online Migration to migrate data between OSS buckets that are owned by multiple accounts, located within the same region, or located across multiple regions.

Data Online Migration allows you to migrate data between buckets in any of the following conditions:

- Buckets are located within the same region.
- Buckets are located across multiple regions.
- Buckets belong to different Alibaba Cloud accounts.

To use Data Online Migration, you need only to log on to the Data Transport console, specify information about the source and destination buckets, and create a migration job. After starting a migration job, you can perform management tasks for the job through the console. For example, you can view the migration progress and bandwidth throttling of the job. Additionally, you can use the console to generate a migration report to view the list of migrated files and the list of files that failed to be migrated.

For more information, see Migrate data between Alibaba Cloud Object Storage Service (OSS) buckets.

1.2 Migrate data sources from a third party to OSS

This topic describes how to use Alibaba Cloud Data Online Migration to migrate data sources from a third party to Alibaba Cloud OSS.

To use Data Online Migration, you need only to log on to the Data Transport console, specify information about the source and destination buckets, and create a migration job. After starting a migration job, you can perform management tasks for the job through the console. For example, you can view the migration progress and bandwidth throttling of the job. Additionally, you can use the console to generate a migration report to view the list of migrated files and the list of files that failed to be migrated.

For more information, see the following documents:

- Migrate data from HTTP/HTTPS sources to OSS
- Migrate data from Tencent Cloud Object Service (COS) to OSS
- Migrate data from Amazon Simple Storage Service (Amazon S3) to OSS
- Migrate data from Qiniu Cloud's object storage (KODO) to OSS
- Migrate data from Azure Blob Storage to OSS
- Migrate data from UPYUN Storage Service (USS) to OSS
- Migrate data from Baidu Object Storage (BOS) to OSS
- Migrate data from Kingsoft Standard Storage Service (KS3) to OSS
- Migrate data from ECS instances to OSS
- Migrate data from NAS to OSS

1.3 Migrate data from Amazon S3 to Alibaba Cloud OSS

OSS provides S3 API compatibility that allows seamless migration of data from Amazon S3 to Alibaba Cloud OSS. After data is migrated from Amazon S3 to OSS, you can still use S3 APIs to access OSS. You only need to configure your S3 client application as follows:

1. Acquire the AccessKeyId and AccessKeySecret of your OSS primary account and sub-account, and configure the acquired AccessKeyID and AccessKeySecret in the client and SDK you are using.
2. Configure the endpoint for client connection to OSS endpoint. For more information, see Regions and endpoints.

Migration procedures

For details about migration procedures, see Use OssImport to migrate data.

Use S3 APIs to access OSS after migration

Take note of the following when you use S3 APIs to access OSS after the migration from S3 to OSS.

- Path style and virtual hosted style

  Virtual hosted style supports accessing OSS by placing the bucket into the host header. For security reasons, OSS only supports virtual hosted style access. Therefore, configurations on your client application are required after the migration from S3 to OSS. Some S3 tools use path style access by default, which also requires proper configurations. Otherwise, OSS may report errors and prohibit access.

- Permission definitions in OSS are not quite the same as they are in S3. You may adjust the permissions as necessary after the migration. See the following table for the main differences between the two.

Note:

- See OSS access for more information on the differences.
- OSS supports only three canned ACL modes in S3: private, public-read, and public-read-write.

Items	Amazon S3 permissions	Amazon S3	Alibaba Cloud OSS
Bucket	READ	With the List permission on the bucket	For all objects under the bucket , if no object permission is set for an object, the object is readable.
	WRITE	Objects in the bucket are writable or overwritable.	- Writable for objects not existing under the bucket. - If no object permission is set for an object existing in the bucket , the object is overwritable. - Initiate multipart upload is allowed.

Items	Amazon S3 permissions	Amazon S3	Alibaba Cloud OSS
	READ_ACP	Read bucket ACLs.	Read bucket ACLs . Only the bucket owner and the authorized sub-account have the permission of reading bucket ACLs.
	WRITE_ACP	Configure bucket ACLs.	Configure bucket ACLs. Only the bucket owner and the authorized sub-account have the permission of configuring bucket ACLs.
Object	READ	Objects are readable.	Objects are readable.
	WRITE	N/A	Objects are overwritable.
	READ_ACP	Read object ACLs.	Read object ACLs . Only the bucket owner and the authorized sub-account have the permission of reading object ACLs.
	WRITE_ACP	Configure object ACLs.	Configure object ACLs. Only the bucket owner and the authorized sub-account have the permission of configuring object ACLs.

- Storage classes

  OSS supports the Standard, IA, and Archive storage classes, which correspond to STANDARD, STANDARD_IA, and GLACIER respectively in Amazon S3. You can convert the storage class of your OSS object as needed.

  Different from Amazon S3, OSS does not support specifying the storage class directly when uploading an object. The storage class of the object is determined by that of the bucket. OSS supports three bucket storage classes: Standard, IA, and Archive. You can use the lifecycle rules to automatically transition objects between storage classes.

  To read an Archive object in OSS, restore it first by initiating a restore request. Different from S3, OSS does not allow setting the lifetime of the restored (active) copy. Therefore, OSS ignores the lifetime (Days) set in the S3 API. The restored state lasts for one day by default, and can be prolonged to seven days at most. After that, the object enters the frozen state again.
- ETag
  - For the object uploaded by using a PUT request, the ETag of an OSS object and that of an Amazon S3 object differ in case sensitivity. The ETag is in upper case for an OSS object and in lower case for an S3 object. If your client has ETag validation, ignore case.
  - For the objects uploaded by Multipart Upload, OSS takes the ETag calculation method that is different from S3.

Compatible S3 APIs

- Bucket operations:
  - Delete Bucket
  - Get Bucket (list objects)
  - Get Bucket ACL
  - Get Bucket lifecycle
  - Get Bucket location
  - Get bucket Logging
  - Head Bucket
  - Put Bucket
  - Put Bucket ACL
  - Put Bucket lifecycle
  - Put Bucket logging
- Object operations:
  - Delete Object
  - Delete Objects
  - Get Object
  - Get Object ACL
  - Head Object
  - Post Object
  - Put Object
  - Put Object Copy
  - Put Object ACL
- Multipart operations:
  - Abort Multipart Upload
  - Complete Multipart Upload
  - Initiate Multipart Upload
  - List Parts
  - Upload Part
  - Upload Part Copy

1.4 Use ossimport to migrate data

This topic describes how to migrate data from third-party storage products (or from another OSS source) to OSS by using ossimport.

Environment configuration

ossimport can be deployed in two modes: standalone mode and distributed mode.

- Standalone mode applies to small-scale data migration scenarios where the data size is smaller than 30 TB.
- Distributed mode applies to large-scale data migration scenarios.

For example, you need to migrate 500 TB of data from an AWS S3 bucket in the Tokyo region to an OSS bucket in the China East 1 (Hangzhou) region within a week. Before migrating the data, you must configure environments to deploy ossimport in distributed mode as follows:

- Activate OSS.
  1. Use your Alibaba Cloud account to create an OSS bucket in China East 1 (Hangzhou).
  2. Create a RAM user in the RAM console, and then grant OSS access permission s to the RAM user. You also need to securely store the AccessKeyID and AccessKeySecret of the RAM user.
- Purchase ECS instances.

  Purchase ECS instances with two CPUs and 4 GB of memory in the China East 1 (Hangzhou) region (in which the OSS bucket is also created). If you want to release the ECS instances after data migration, we recommend that you select Pay-As-You-Go as the billing method when purchasing the instances.

  The number of required ECS instances can be calculated as follows: X/Y/(Z/100). In the formula, X indicates the size of data that needs to be migrated, Y indicates the number of days that the migration requires, and Z indicates the transfer speed of a single ECS instance (MB/s), that is, how much TB of data can be migrated by a single ECS instance each day (calculated as Z/100). Assume that the transfer speed of an ECS instance is 200 MB/s (that is, an ECS instance can migrate 2 TB of data each day). This means you must purchase 36 ECS instances in total (calculated from 500/7/2).

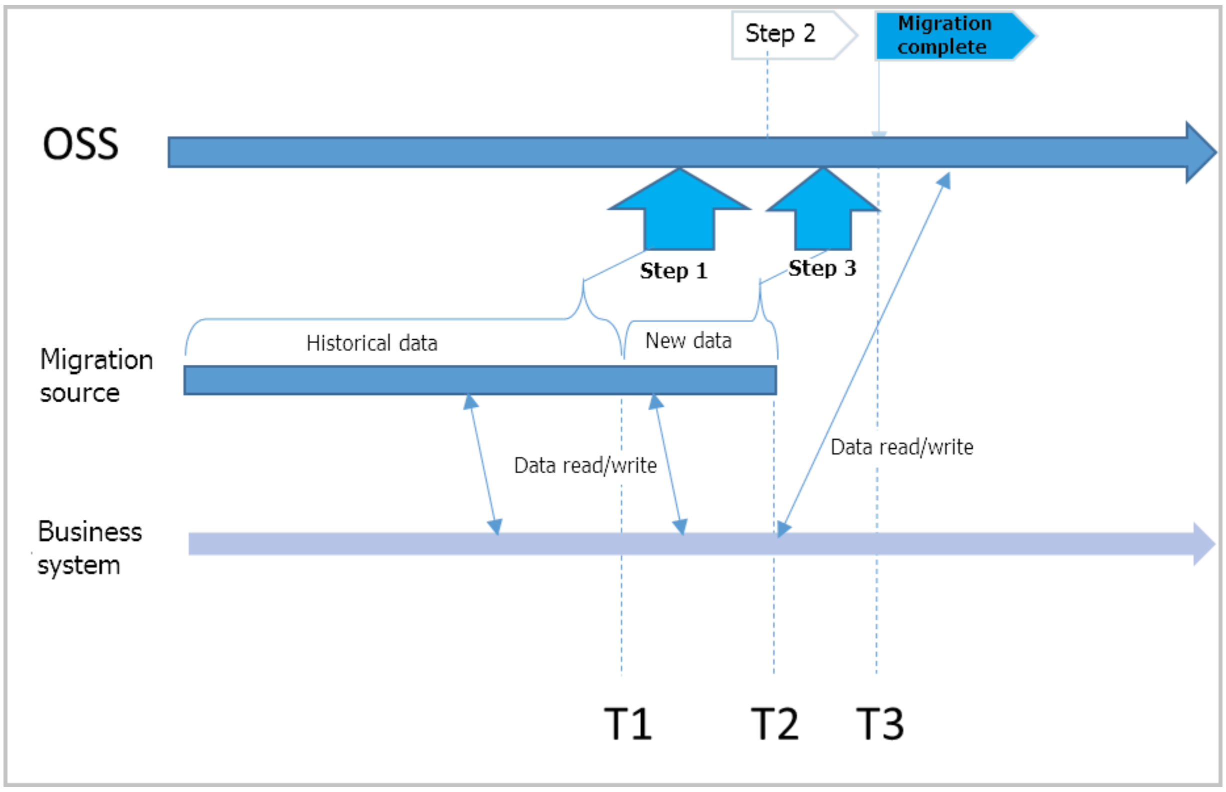
- Configure ossimport

  For the large-scale data migration requirement in this example, you must deployed ossimport in ECS instances in distributed mode. For the configuration information about distributed mode, such as `conf / job . cfg` , `conf / sys . properties` , and concurrency control, see Architecture and configuration. For more information about ossimport distributed deployment, such as the downloading method of ossimport and the troubleshooting of ossimport configuration, see Distributed deployment.

Procedure

You can use ossimport in distributed mode to migrate data from AWS S3 to OSS as follows:

Note:
After deploying ossimport in distributed mode in the ECS instances, use ossimport to download data from the AWS S3 bucket in the Tokyo region to the ECS instances in China East 1 (Hangzhou). We recommend you download the data through Internet. Use ossimport to upload the data from the ECS instances to the OSS bucket in China East 1 (Hangzhou). We recommend you upload the data through the intranet.



1. Fully migrate the historical data in AWS S3 before the time T1. For more information, see the Running section in Distributed deployment.

   T1 is a UNIX timestamp, namely, the number of seconds elapsed since UTC 00:00, January 1, 1970, and can be obtained by running the `date +% s` command).
2. In the OSS console, enable Back-to-Origin for the target bucket and set the access URL of the AWS S3 as the origin URL.
3. Switch all read and write operations on AWS S3 to OSS, and record the time (T2).

   In this way, all historical data before T1 is directly read from the OSS bucket, and the data stored between T1 and T2 is read from AWS S3 through the mirroring back -to-origin function of OSS.

   After T2, all new data is written to OSS and no new data is written to AWS S3.
4. Modify the item `importSince = T1` in the configuration file `job.cfg`, and then start a migration task again to migrate the data added between T1 and T2.

   Note:

   - After step 4, all read and write operations are performed on the target OSS bucket. Data stored in AWS S3 is historical data, which can be retained or deleted as needed.
   - ossimport only migrates and verifies data but does not delete it.

Various costs are incurred during data migration, including the cost of ECS instances , traffic costs, storage costs, and time-dependent costs. Furthermore, if the size of the data to be migrated is larger than 1 TB, the storage cost increases due to the time required for the migration. However, the storage cost generally remains lower than the costs associated with network traffic and ECS instances. You can reduce the time needed for migration by using more ECS instances.

References

For more information about ossimport, see the following documentations:

Distributed deployment

Architecture and configuration

FAQ

1.5 Back up an HDFS to OSS for disaster tolerance

Background

Currently, many data centers are constructed using Hadoop, and in turn an increasing number of enterprises want to smoothly migrate their services to the cloud.

Object Storage Service (OSS) is the most widely-used storage service on Alibaba Cloud . The OSS data migration tool, ossimport2, allows you to sync files from your local devices or a third-party cloud storage service to OSS. However, ossimport2 cannot read data from Hadoop file systems. As a result, it becomes impossible to make full use of the distributed structure of Hadoop. In addition, this tool only supports local files. Therefore, you must first download files from your Hadoop file system (HDFS) to your local device and then upload them using the tool. This process consumes a great deal of time and energy.

To solve this problem, Alibaba Cloud’ s E-MapReduce team developed a Hadoop data migration tool emr-tools. This tool allows you to migrate data from Hadoop directly to OSS.

This chapter introduces how to quickly migrate data from HDFS to OSS.

Prerequisites

Make sure your current machine can access your Hadoop cluster. That is, you must be able to use Hadoop commands to access HDFS.

```
hadoop  fs  - ls  /
```

Migrate Hadoop data to OSS

1. Downloademr-tools.

   Note:
   emr-tools is compatible with Hadoop versions 2.4.x, 2.5.x, 2.6.x, and 2.7.x.
2. Extract the compressed tool to a local directory.

   ```
   tar  jxf  emr - tools . tar . bz2
   ```
3. Copy HDFS data to OSS.

   ```
   cd  emr - tools
   ```

```
./ hdfs2oss4e  mr . sh  / path / on / hdfs   oss :// accessKeyI  d :
 accessKeyS  ecret @ bucket - name . oss - cn - hangzhou . aliyuncs
 . com / path / on / oss
```

The relevant parameters are described as follow.

Parameters	Description
accessKeyId	The key used to access OSS APIs. For more information, see How to obtain AccessKeyId and AccessKeyS ecret.
accessKeySecret	
bucket-name.oss-cn-hangzhou.aliyuncs .com	The OSS access domain name, including the bucket name and endpoint address.

The system enables a Hadoop MapReduce task (DistCp).

4. After the task is completed, local data migration information is displayed. This information is similar to the following sample.

```
17 / 05 / 04   22 : 35 : 08   INFO   mapreduce . Job :  Job
job_149380  0598643_00  09   completed   successful  ly
17 / 05 / 04   22 : 35 : 08   INFO   mapreduce . Job :  Counters :
38
 File   System   Counters
          FILE :  Number   of   bytes   read = 0
          FILE :  Number   of   bytes   written = 859530
          FILE :  Number   of   read   operations = 0
          FILE :  Number   of   large   read   operations = 0
          FILE :  Number   of   write   operations = 0
          HDFS :  Number   of   bytes   read = 263114
          HDFS :  Number   of   bytes   written = 0
          HDFS :  Number   of   read   operations = 70
          HDFS :  Number   of   large   read   operations = 0
          HDFS :  Number   of   write   operations = 14
          OSS :  Number   of   bytes   read = 0
          OSS :  Number   of   bytes   written = 258660
          OSS :  Number   of   read   operations = 0
          OSS :  Number   of   large   read   operations = 0
          OSS :  Number   of   write   operations = 0
 Job   Counters
          Launched   map   tasks = 7
          Other   local   map   tasks = 7
          Total   time   spent   by   all   maps   in   occupied
slots  ( ms )= 60020
          Total   time   spent   by   all   reduces   in   occupied
  slots  ( ms )= 0
          Total   time   spent   by   all   map   tasks  ( ms )=
30010
          Total   vcore - millisecon  ds   taken   by   all   map
tasks = 30010
          Total   megabyte - millisecon  ds   taken   by   all   map
  tasks = 45015000
 Map - Reduce   Framework
          Map   input   records = 10
          Map   output   records = 0
```

```
            Input   split   bytes = 952
            Spilled   Records = 0
            Failed   Shuffles = 0
            Merged   Map   outputs = 0
            GC   time   elapsed  ( ms )= 542
            CPU   time   spent  ( ms )= 14290
            Physical   memory  ( bytes )  snapshot = 1562365952
            Virtual   memory  ( bytes )  snapshot = 1731742105  6
            Total   committed   heap   usage  ( bytes )= 1167589376
    File   Input   Format   Counters
            Bytes   Read = 3502
    File   Output   Format   Counters
            Bytes   Written = 0
    org . apache . hadoop . tools . mapred . CopyMapper $ Counter
            BYTESCOPIE  D = 258660
            BYTESEXPEC  TED = 258660
            COPY = 10
  copy   from  / path / on / hdfs   to   oss :// accessKeyI  d :
  accessKeyS  ecret @ bucket - name . oss - cn - hangzhou . aliyuncs
  . com / path / on / oss   does   succeed  !!!
```

5. You can use osscmd to view information about OSS data.

```
osscmd   ls   oss :// bucket - name / path / on / oss
```

Migrate OSS data to Hadoop

If you have already created a Hadoop cluster on Alibaba Cloud, you can use the following command to migrate data from OSS to the new Hadoop cluster.

```
./ hdfs2oss4e  mr . sh   oss :// accessKeyI  d : accessKeyS  ecret @
 bucket - name . oss - cn - hangzhou . aliyuncs . com / path / on /
 oss  / path / on / new - hdfs
```

More scenarios

In addition to offline clusters, you can also use emr-tools for Hadoop clusters constructed on ECS. This allows you to quickly migrate a self-built cluster to the E-MapReduce service.

If your cluster is already on ECS, but in a classic network, it will not provide good interoperability with services in Virtual Private Cloud (VPC). In this case, migrate the cluster to a VPC instance. Follow these steps to migrate the cluster:

1. Use emr-tools to migrate data to OSS.
2. Create a new cluster (create it yourself or use E-MapReduce) in the VPC environment.
3. Migrate data from OSS to the new HDFS cluster.

If you use E-MapReduce, on the Hadoop cluster, you can directly access OSS using Spark, MapReduce and Hive. This not only avoids one data copy operation (from OSS to HDFS), but also greatly reduces storage costs. For more information about cost reduction, see EMR+OSS: Separated storage and computing.

2 Upload data to OSS through Web applications

2.1 Use PostObject to upload data to OSS through Web applications

2.1.1 Overview of direct transfer on Web client

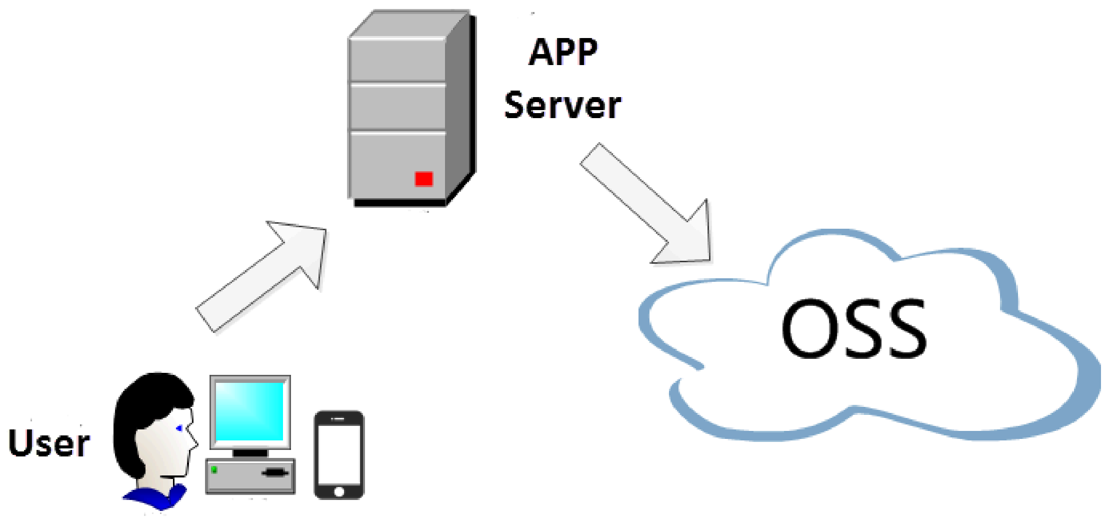
Purpose

This document with the help of two examples, elaborates how to transfer a file in HTML form directly to OSS.

- Example 1: Describes how to add a signature on a server (PHP) and then upload the file directly to OSS using a form.
- Example 2: Describes how to add a signature on the server (PHP), and set the callback upon uploading on the server. Then, upload the form directly to OSS. After that, OSS calls back the application server and returns the result to the user.

Background

Every OSS user may use the upload service. This is because, the data is uploaded using Web pages and it includes some HTML5 pages in some apps. The demand to upload these services is strong. Many users choose to upload files to the application servers through browsers/apps, and then the application server uploads the files to OSS.



However, the preceding method has following limitations:

- Low uploading speed: Intially, files are uploaded to the application server, and then to OSS. Therefore, the workload of transmission over the Internet is doubled . If the data is transferred directly to OSS without passing through the application server, the speed increases significantly. Moreover, OSS uses BGP bandwidth, thus ensuring a high speed for operators in different places.
- Poor scalability: As the number of users increases in future, the application server may constitute a bottleneck.
- High cost: The traffic consumed for uploading files directly to OSS is free of charge . If data is uploaded directly to OSS without passing through the application server , the costs of several application servers can be saved.

Basic

The application server uses PHP script language to return the signature. Click here for the example.

Advanced

The application server returns the signature using the PHP script language and implements uploading callback. Click here for the example.

2.1.2 Javascript client signature pass-through

Backdrop

The client is signed directly with JavaScript and then uploaded to OSS. Please see the background introduction in the Web-side direct transmission practice.

Example

The following is a description of the plupoad. An example of signing on the Javascript side and then passing data directly to OSS.

Use your phone to test if the upload is valid. Two-dimensional code: can use mobile phone (WeChat, QQ, mobile phone browser, etc) give it a try. (This is not an advertisement, but a two-dimensional code for the above-mentioned web site, in order to let everyone see this implementation can run perfectly on the mobile phone .)

File Upload is a public upload to a test. Bucket is cleaned up regularly, so don't pass on sensitive and important data.

code download

Click here: oss-h5-upload-js-direct.zip

Principle

- The functionality of this example
  - Direct submission of form data (that is, postobject) to OSS using procopad.
  - Support HTML5, Flash, Silverlight, html4 and other protocols upload.
  - Can run in PC browser, mobile phone browser, WeChat, etc.
  - You can choose to upload multiple files.
  - Display upload progress bar.
  - You can control the size of the uploaded file.
  - You can set the upload to the specified directory and set whether the Upload File name is a random file name or a local file name.

  The Postobject API details of OSS can be referred.
- Plupload

  Procopad is a simple, easy-to-use and powerful file upload tool, Supports multiple upload methods, including HTML5, Flash, Silverlight, Html4. Intelligent Detection of the current environment, choosing the best way to do so, and adopting HTML5 is a priority.
- Key code

  Because the OSS supports the post protocol. So all you need to do is bring the OSS signature when you send a POST request. The core code is as follows:

```
VaR   deliader  =  new   pluopad .  deliader  ({
    Runtimes : ' html5 ,  Flash ,  Silverligh  t ,  html4  ',
    Browse_but  ton : ' selectedfi  le  ',
   //  Runtimes : ' Flash  ',
    Container :  Document .  getelindby  id  (' container  '),
    Flash_swf_  url : " lib / plupload - 2 . 1 . 2 / JS / Moxie .
swf  ',
    Silverligh  t_xap_url : ' lib / FIG / JS / moxie .  xap  ',
    URL :  host ,
    Multipart_  params  :{
       ' Filename  ':' $ { filename  }',
       ' Key ': '$ { filename  }',
       ' Policy ':  policybase  64 ,
       ' Porter ':  Access   Sid ,
       ' Success   _   action_sta  tus ': ' 200 ', //  Let   the
server   return   200 ,  otherwise ,  204   is   returned   by
default
       ' Signature  ':  Signature ,
   },
 ....
}
```

One thing to note here is 'filename ': '$ {Filename }', The purpose of this piece of code is to indicate that the original file text is maintained after the upload. If you want to upload to a specific directory such as ABC, the file name remains the same as the original file name, so this should be written:

```
Multipart_  params  :{
       ' Filename  ': ' abc /' + '$ { filename  }',
       ' Key ': '$ { filename  }',
       ' Policy ':  policybase  64 ,
       ' Porter ':  Access   Sid ,
       ' Success   _   action_sta  tus ': ' 200 ', //  Let   the
server   return   200 ,  otherwise ,  204   is   returned   by
default
       ' Signature  ':  Signature ,
   },
```

- Set to random file name

Sometimes you need to set the user-uploaded file to a random file name, And the suffix is consistent with the client file. In the example, two radios are used to distinguish, If you want to be fixed to a random file name at the time of delivery, you can change the function to the following:

```
 Function   fig  (){
     G_object_n  ame_type  = ' random   _   name  ';
}
```

If you want to fix the file that is set to the user when you pass it over, you can change the function:

```
 Function   fig  (){
     G_object_n  ame_type  = ' Local   _   name  ';
}
```

- Set the upload directory

Files can be uploaded to the specified directory, and directory-related settings can be experienced in the example, if you want your code to be uploaded to a fixed directory such as ABC, you can change it as follows, note '/' End.

```
 Function   get_dirnam  e  ()
{
     G_dirname  = " ABC  /";
}
```

- Upload signatures

Signing signature is primarily a signing of policytext, and the simplest example is:

```
 VaR   policytext  = {
    " Expiration ": " 00 :  00 :  00 . 000z  ", //  set   the
 failure   time   for   this   policy , after   which   the
 failure   time   is   exceeded , there ' s   no   way   to
 upload   files   through   this   policy .
    " Conditions  ":[
    [" Content - Length - range ",  0 ,  1048576000 ] //  set   the
   size   limit   for   the   uploaded   file , if   this   size
 is   exceeded , the   file   was   uploaded   to   OSS , and   it
   will   be   reported   wrong .
    ]
}
```

- Cross Domain CORS

  Note:
  Make sure that the bucket property CORS setting supports the POST method. Because this HTML is uploaded directly to OSS, cross-domain requests are generated. You must set the allow cross-domain inside the bucket property.

  Set the following figure:

  Note:
  In earlier versions of the IE browser, pluopad is executed in flash mode. Crossdomain. xml must be set , The setting method can be referred to: click here

- CAUTION

  Writing the access key ID and the access key secret inside the code poses a risk of leakage. It is recommended to use back-end signatures upload scheme: Post-server signatures after direct transmission of web pages

2.1.3 Direct transfer after adding a signature on the server

Background

Direct signature by JS clients has a serious hidden security hazard in that the OSS AccessId/AcessKey are exposed on the frontend which may be accessible to others . This document explains how to get a signature from and upload a policy to the backend PHP code.

The logic for uploading a signature to the backend is as follows:

1. Before uploading an image, the client obtains the uploaded policy and signature from the application server.
2. The client directly uploads the obtained signature to the OSS.

Signature sample uploaded to the backend

- Download sample:
  - Click here to download a test sample on a PC browser.
  - You can test whether the upload was effective on a mobile phone. You can use a mobile phone app (such as WeChat, QQ, and mobile browsers) to scan the QR code.

    This is not an advertisement, but a QR code for the preceding URL. This operation allows you to see whether the service works as intended on mobile phones.
- Download code:

  Click here to download the code.

  This example adopts the backend signature, and uses PHP language.

  - Click here for the example of a backend signature using Java language.
  - Click here for the example of a backend signature using Python language.
  - Click here for the example of a backend signature using Go language.

  Usage of other languages:

  1. Download the corresponding language example.
  2. Modify the example code. For instance, set the listening port, and then start running.
  3. At `upload . js` in `oss - h5 - upload - js - php . zip`, change the variable *`serverUrl`* to the address configured at step 2. For example, = serverUrl=`http :// 1 . 2 . 3 . 4 : 8080` or serverUrl=`http :// abc . com / post /`.

Principle of constructing a Post signature on the server end

The OSS PostObject method is used for uploads. You can construct a PostObject request in the browser using Plupload and send the request to the OSS. Signatures are implemented on the server in PHP. In the same principle, the server can be compiled in Java, .NET, Ruby, Go, or Python language. The core logic is to construct a Post signature. The Java and PHP examples are provided here. The following steps are required:

1. The webpage requests the signature through JavaScript from the server end.

2. After JavaScript gets the signature, it uploads the signature to the OSS through Plupload.

- Implementation

  1. Populate the fields with your ID, key, and bucket.

     Modify php/get.php:

     - Set the variable $id to AccessKeyId.
     - Set $key to AccessKeySecret.
     - Set $host to bucket+endpoint.

       Note:
       For information on the endpoint, see Basic OSS concepts.

```
$ id = ' xxxxxx ';
  $ key = ' xxxxx ';
  $ host  = ' http :// post - test . oss - cn - hangzhou .
 aliyuncs . com
```

  2. You must set CORS for the bucket to guarantee browser safety.

     Note:
     Make sure that the CORS settings of the bucket attribute support the POST method. This is because, HTML directly uploads data to OSS and produces a cross-origin request in the process. Hence, you must allow cross-original requests in the bucket attributes.

     For procedure, see Set CORS. The settings are as follows:

     Note:
     In earlier-version IE browsers, Plupload is executed in flash. You must set crossdomain.xml.

Details of core logic

- Set random object names

  You often need to name uploaded objects randomly, if they have the same suffix as the objects on the client. In this example, two radios are used to differentiate. If you want to fix the settings to apply random names to the uploaded objects, you can change the function to the following:

```
 function   check_obje  ct_radio () {
     g_object_n  ame_type  = ' random_nam  e ';
}
```

If you want to set uploads to the user's objects, you can change the function to the following:

```
 function   check_obje  ct_radio () {
     g_object_n  ame_type  = ' local_name ';
}
```

- Set the upload directory

  The upload directory is specified by the server end (in PHP), which enhances security. Each client is only allowed to upload objects to a specific directory. This guarantees security by isolation. The following code changes the upload directory address to `abc /` (the address must end with `/`).

```
$ dir  = ' abc /';
```

- Set the filtering conditions for uploaded objects

  We often need to set the filtering conditions for uploads. For example, only allowing image uploads, setting the size of uploaded objects, and disallowing repeated uploads. You can use the filters parameter for this.

```
 var   uploader  =  new   plupload . Uploader ({
    ……
     filters : {
         mime_types  : [ // Only   images   and   zip   objects
 are   allowed   to   be   uploaded
        {  title  : " Image   files ",  extensions  : " jpg , gif ,
 png , bmp " },

        ],
         max_file_s  ize  : ' 400kb ', // Only   objects   with
   a   maximum   size   of   400   KB   are   allowed   to   be
 uploaded
         prevent_du  plicates  :  true  // Repeated   objects   are
   not   allowed   to   be   selected
```

```
    },
```

Use the Plupload attribute filters to set filtering conditions.

Explanations of the preceding setting values:

- mime_types: Restrict extensions of the uploaded objects.
- max_file_size: Restrict the size of the uploaded objects.
- prevent_duplicates: Restrict repeated uploads.

> Note:
> The filter conditions are not required. You can comment out the filtering condition, if you do not need it.

- Get uploaded object names

  If you want to know the name of the uploaded object, you can use Plupload to call the FileUploaded event, as follows:

```
FileUpload  ed :  function ( up ,  file ,  info ) {
            if  ( info . status  ==  200 )
           {
                document . getElement  ById ( file . id ).
getElement  sByTagName (' b ')[ 0 ]. innerHTML  = ' upload   to
oss   success ,  object   name :' +  get_upload  ed_object_  name (
file . name );
           }
            else
           {
                document . getElement  ById ( file . id ).
getElement  sByTagName (' b ')[ 0 ]. innerHTML  =  info . response
;
           }
    }
```

  You can use the following functions to get the names of the objects uploaded to OSS . The file.name property records the names of the uploaded local objects.

```
get_upload  ed_object_  name ( file . name )
```

- Upload signatures

  JavaScript can get the policyBase64, accessid, and signature variables from the backend. The following is the core code for getting the three variables:

```
phpUrl  = './ php / get . php '
        xmlhttp . open ( " GET ",  phpUrl ,  false  );
        xmlhttp . send (  null  );
        var   obj  =  eval  ("(" +  xmlhttp . responseTe  xt +
")");
        host  =  obj [' host ']
        policyBase  64  =  obj [' policy ']
        accessid  =  obj [' accessid ']
```

```
          signature  =  obj [' signature ']
          expire  =  parseInt ( obj [' expire '])
          key  =  obj [' dir ']
```

Parse xmlhttp.responseText (the following only serves as an example. The actual format may vary, but the values of signature, accessid, and policy must exist).

```
{" accessid ":" 6MKOxxxxxx  4AUk44 ",
" host ":" http :// post - test . oss - cn - hangzhou . aliyuncs .
 com ",
" policy ":" eyJleHBpcm  F0aW9uIjoi  MjAxNS0xMS  0wNVQyMDoy
 MzoyM1oiLC  Jjxb25kaXR  pb25zIjpbW  yJjcb250ZW  50LWxlbmd0
 aC1yYW5nZS  IsMCwxMDQ4  NTc2MDAwXS  xbInN0YXJ0  cy13aXRoIi
 wiJGtleSIs  InVzZXItZG  lyXC8iXV19 ",
" signature ":" I2u57FWjTK  qX / AE6doIdyff  151E =",
" expire ": 1446726203 ," dir ":" user - dir /"}
```

- accessid: It is the Accessid of the user request. However, disclosing Accessid does not impact data security.
- host: The domain name to which the user wants to send an upload request.
- policy: A policy for uploading user forms. It is a Base64-encoded string.
- signature: A signature string for the policy variable.
- expire: It is the expiration time of the current upload policy. This variable is not sent to OSS, because it is already indicated in the policy.

Parse policy. The decoded content of the policy is as follows:

```
{" expiration ":" 2015 - 11 - 05T20 : 23 : 23Z ",
" conditions ":[[" content - length - range ", 0 , 1048576000 ],
["  starts - with ","$ key "," user - dir /"]]
```

For more information about Policy, see Policy basic elements.

The key content of the PolicyText specifies the final expiration time of this policy . Before its expiry, this policy may be used to upload objects. Therefore, it is not necessary to obtain a signature from the backend for each upload.

Here, we use the following designs:

- For initial uploads, a signature is obtained for each object upload.
- For subsequent uploads, the current time is compared with the signature time to see whether the signature has expired.
  - If the signature expires, a new signature is obtained.
  - If the signature has not expired, the same signature is used. The expired variable is used here.

The core code is as follows:

```
 now  =  timestamp  =  Date . parse ( new   Date ()) /  1000 ;
[ color =# 000000 ]// This   determines   whether   the   time
 specified   by   the   expire   variable   is   earlier   than
   the   current   time .  If   so ,  a   new   signature   is
 obtained .  3s   is   the   buffer   duration .[/ color ]
     if  ( expire  <  now  +  3 )
{
      .....
     phpUrl  = './ php / get . php '
     xmlhttp . open ( " GET ",  phpUrl ,  false  );
     xmlhttp . send (  null  );
      ......
}
 return  .
```

We see that starts-with has been added to the policy content. This indicates the name of the object to be uploaded must start with the user-dir (this string can be customized).

This setting is added because, in many scenarios, one bucket is used for one app and contains the data of different users. To prevent the data from being overwritte n, a specific prefix is added to the objects uploaded by a specific user to OSS.

However, an issue occurs. Once the users obtains this policy, they can modify the upload prefix before the expiration time to upload objects to another user's directory. To resolve this issue, you can set the application server to specify the prefix of the uploaded objects by a specific user at the time of upload. In this case

, no one can upload objects with another user's prefix even after obtaining the policy. This guarantees data security.

Summary

In the sample mentioned in this document, the webpage end requests the signature from the server end during uploads from the webpage end, and then objects are uploaded directly, with no pressure on the server end. This approach is safe and reliable.

However in this sample, the backend program is not immediately aware of the number or identity of objects uploaded. You can use upload callback to see which objects were uploaded. This sample cannot implement multipart and breakpoint.

Related Documents

- Basic concepts
- Set Cross-Origin Resource Sharing (CORS)
- Overview of direct transfer on Web client
- Directly add a signature on the server, transfer the file, and set upload callback
- Set up direct data transfer for mobile apps

2.1.4 Directly add a signature on the server, transfer the file, and set upload callback

Background

For the background information, see Overview of direct transfer on Web client .

The usage of Direct transfer after adding a signature on the server solution experiences a few issues. Once the user uploads data, the application server has to be updated with the files user uploads, the file names, image size (if any images are uploaded), and so on. Hence, the upload callback function is developed for OSS.

- User request logic

1. The user obtains the upload policy and callback settings from the application server.
2. The application server returns the upload policy and callback settings.
3. The user sends a file upload request directly to OSS.
4. Once the file data is uploaded and before a response is sent by OSS to the user, OSS sends a request to the user’s server based on the user’s callback settings.
5. If the server returns success, OSS returns success to the user. If the server returns failed, OSS returns failed to the user. This makes sure the application server is be notified of all images that the user has successfully uploaded.
6. The application server returns information to OSS.
7. OSS returns the information returned by the application server to the user.

In brief, the user needs to upload a file to the OSS server. And, it is assumed that the user’s application server is notified once the upload is completed. In this case, a callback function is required to be set to update user’s application server . Due to this, OSS starts the upload once it receives user’s upload request. It does not return the result to the user directly after uploading, but notifies the user’s application server first with a system-generated message such as “I completed uploading”; then, the application server notifies OSS by sending “I got it. Please pass on the information to my owner” message. After sending these notifications, OSS transfers the result to the user.

- Example

Sample test of user's Computer Browser: Click here to experience the upload callback example

Use your phone to test if the upload is valid. You can use a cell phone (WeChat, QQ, mobile browsers, and so on) to scan the QR code (this is not an advertisement, but the QR code for the URL provided above. The operation allows you to see whether the services work perfectly on cell phones.)

- Download code

  Click here to download the code.

  The example adopts a backend signature and uses PHP language.

  - Click here for the example of a backend signature using Java language.
  - Click here for the example of a backend signature using Go language.
  - Click here for the example of a backend signature using Python language.

  Usage of other languages:

  1. Download the corresponding language example.
  2. Modify the example code, for example, set the listening port, and then start running.
  3. At upload.js in `oss - h5 - upload - js - php - callback . zip`, change the variable severUrl to the address configured at step 2. For example, severUrl = `http :// 1 . 2 . 3 . 4 : 8080` or serverUrl= `http :// abc . com / post /`.
- Quick start guide

  Follow the steps to upload a file to OSS through the Webpage, and OSS sends a callback notification to the application server set by the user.

  1. Set your own id, key, and bucket.

     Setting method: Modify `php / get . php`, and set the variable $id to AccessKeyId, $key to AccessKeySecret, and $host to bucket+endpoint.

     ```
     $ id = '******';
      $ key = ' xxxxx ';
      $ host  = ' http :// post - test . oss - cn - hangzhou . aliyuncs
      . com
     ```
  2. To guarantee browsing security, CORS must be set for bucket.
  3. Set your own callback URL. It is also known as your own callback server address. For example, `http :// abc . com / test . html` (can be accessed through public network). OSS sends the file uploading information to the application server through the callback URL (`http :// abc . com / test`

`. html`) set by you after the file is uploaded. Setting method: Modify php/get.php (for this callback server code instance, see the following content).

```
$ callbackUr  l  = " http :// abc . com / test . html ";
```

For more information such as uploading signature and setting a random file name, click here for uploading details.

- Core code analysis

  The following content is to be added to the code:

```
 new_multip  art_params  = {
& nbsp ;& nbsp ;& nbsp ;& nbsp ; ' key ' :  key  + '${ filename }',
& nbsp ;& nbsp ;& nbsp ;& nbsp ; ' policy ':  policyBase  64 ,
& nbsp ;& nbsp ;& nbsp ;& nbsp ; ' OSSAccessK  eyId ':  accessid ,
& nbsp ;& nbsp ;& nbsp ;& nbsp ; ' success_ac  tion_statu  s ' : '
 200 ', // Instructs   the   server   to   return   200 .  Otherwise
 ,  the   server   returns   204   by   default .
& nbsp ;& nbsp ;& nbsp ;& nbsp ; ' callback ':   callbackbo  dy ,
& nbsp ;& nbsp ;& nbsp ;& nbsp ; ' signature ':  signature ,
 };
```

  The preceding callbackbody is returned by the PHP server. In this example, the following content is obtained by running the PHP scripts on the backend:

```
{" accessid ":" 6MKO ****** 4AUk44 ",
" host ":" http :// post - test . oss - cn - hangzhou . aliyuncs .
 com ",
" policy ":" eyJleHBpcm  F0aW9uIjoi  MjAxNS0xMS  0wNVQyMDo1  M
 ****** iLCJjdb25k  aXRpb25zIj  pbWyJjdb25  0ZW50LWxlb  md0aC1yYW5
 nZSIsMCwxM  DQ4NTc2MDA  wXSxbInN0Y  XJ0cy13aXR  oIiwiJGtle
 SIsInVzZXI  tZGlyXC8iX  V19 ",
" signature ":" VsxOcOudx ****** z93CLaXPz + 4s =",
" expire ": 1446727949 ,
" callback ":" eyJjYWxsYm  Fja1VybCI6  Imh0dHA6Ly  9vc3MtZGVt
 by5hbGl5dW  5jcy5jdb20  6MjM0NTAiL  CJjYWxsYmF  ja0hvc3QiO
 iJvc3MtZGV  tby5hbGl5d  W5jcy5jdb2  0iLCJjYWxs  YmFja0JvZH
 kiOiJmaWxl  bmFtZT0ke2  9iamVjdH0m  c2l6ZT0ke3  NpemV9Jm1p
 bWVUeXBlPS  R7bWltZVR5  cGV9JmhlaW  dodD0ke2lt  YWdlSW5mby
 5oZWlnaHR9  JndpZHRoPS  R7aW1hZ2VJ  dbmZvLndpZ  HRofSIsImN
 hbGxiYWNrQ  m9keVR5cGU  iOiJhcHBsa  WNhdGlvbi9  4LXd3dy1mb
 3JtLXVybGV  uY29kZWQif  Q ==","dir ":" user - dirs /"}
```

  The preceding callbackbody is the Base64 encoded callback content in the returned results.

  The decoded content is as follows:

```
{" callbackUr  l ":" http :// oss - demo . aliyuncs . com : 23450 ",
" callbackHo  st ":" oss - demo . aliyuncs . com ",
" callbackBo  dy ":" filename =${ object }& size =${ size }&
 mimeType =${ mimeType }& height =${ imageInfo . height }& width =${
 imageInfo . width }",
" callbackBo  dyType ":" applicatio  n / x - www - form - urlencoded
 "}
```

Content analysis:

- callbackUrl: Specifies the URL request sent by OSS to this host.
- callbackHost: Specifies the Host header to be included in the request header when this request is sent by the OSS.
- callbackBody: Specifies the content sent to the application server upon OSS request. This can include a file name, size of the file, type, and image and its size (if any).
- callbackBodyType: Specifies the Content-Type requested to be sent.

- Callback application server

  Step 4 and 5 is important in the user’ s request logic. When OSS interacts with the application server. The following are a few questions explained with answers.

  - Question: If I am a developer, how can I confirm that the request was sent from OSS?

    Answer: When OSS sends a request, it constructs a signature with the applicatio n server. Both use signatures to guarantee security.
  - Question: How is this signature constructed? Is there any sample code?

    Answer: Yes. The preceding example shows the sample code of the application server callback: `http :// oss - demo . aliyuncs . com : 23450` (only supports Linux now).

    The preceding code runs as follows: callback_app_server.py.zip

    Running solution: Directly run the file *python callback_a pp_server . py* under the Linux system.

    The program automatically implements a simple http server. To run this program, you may need to install the system environment on which the RSA depends.
  - Question: Why the callback request received by my application server does not have an Authorization header?

    Answer: Some Web servers resolve the Authorization header automatically, for example, apache2. Therefore, it is set not to resolve this header. Using apache2 as an example, the specific setting method is as follows:

    1. Start the rewrite module, and run the command: `a2enmod rewrite .`
    2. Modify the configuration file */ etc / apache2 / apache2 . conf* (it varies with the installation path of apache2). Set Allow Override to All, and then add the following content:

       ```
       RewriteEng ine on
       ```

```
RewriteRul  e  .* - [ env = HTTP_AUTHO  RIZATION :%{ HTTP :
Authorizat  ion }, last ]
```

The sample program demonstrates how to check the signature received by the application server. You must add the code for parsing the format of the callback content received by the application server.

Callback application server versions in other languages

- Java version:
  - Download address: click here
  - Running method: Extract the archive and run `java  - jar  oss - callback - server - demo . jar  9000` (9000 is the port number and can be changed as required).

    Note:
    This jar runs on java 1.7. If any issue occurs, you may make changes based on the provided code. This is a maven project.
- PHP version:
  - Download address: click here
  - Running method: Deploy a program to an Apache environment. Due to the characteristics of PHP language, retrieving headers depends on the environment. You can make changes to the example based on your own environment.
- Python version:
  - Download address: click here
  - Running method: Extract the archive and directly run python callback_a pp_server.py. The program implements a simple HTTP server. To run this program, you may be required to install the system environment on which the RSA depends.
- Ruby version:
  - Download address: click here
  - Running method: ruby aliyun_oss_callback_server.rb.

Summary

- Example 1: Describes how to add a signature directly on the JavaScript client and upload a file in the form to OSS directly. oss-h5-upload-js-direct.tar.gz
- Example 2: Describes how to obtain a signature from the backend using the PHP script and then upload the file in a form to OSS directly. oss-h5-upload-js-php.tar.gz
- Example 3: Describes how to obtain a signature from the backend using the PHP script, and perform callback after uploading, and then, upload the form directly to OSS. Consequently, OSS calls back the application server and returns the result to the user. oss-h5-upload-js-php-callback.tar.gz

3 Application server

3.1 Set up direct data transfer for mobile apps

Background

In the era of mobile Internet, mobile apps upload more and more data every day. By handing off their data storage issues to OSS, developers can focus more on their app logic.

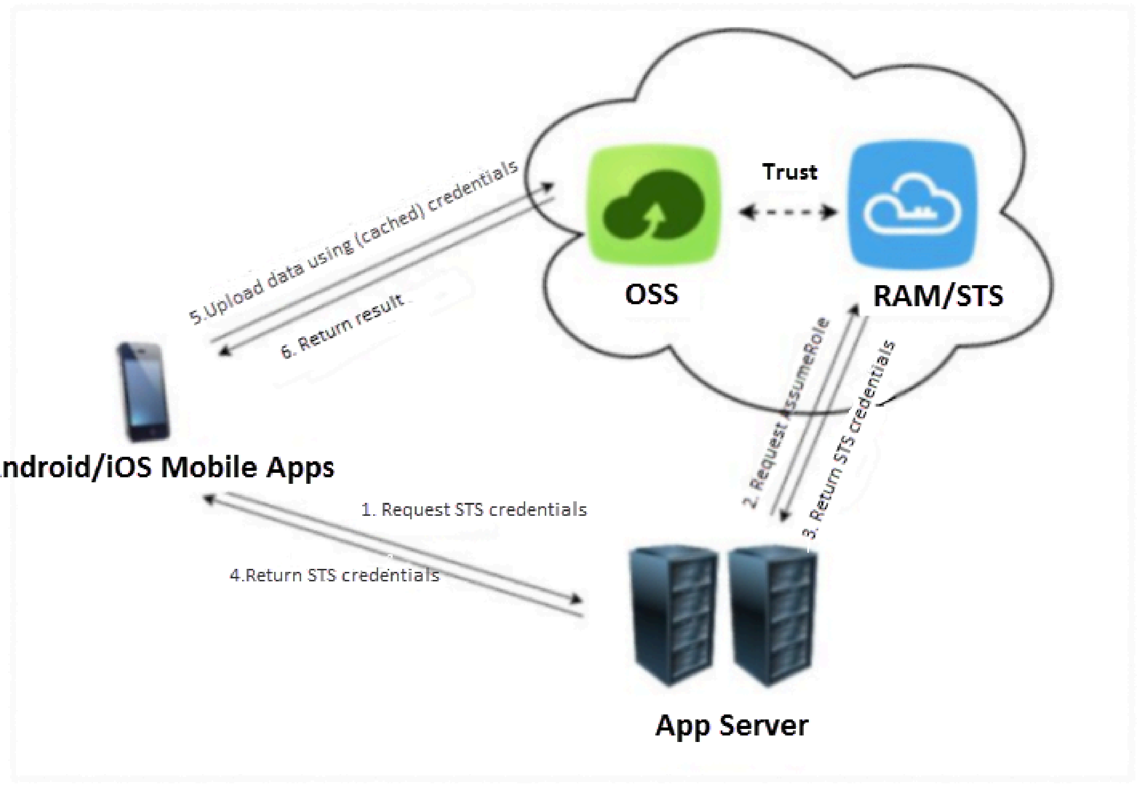
This article describes how to set up an OSS-based direct data transfer service for a mobile app in 30 minutes. Direct data transfer is a service that allows a mobile app to directly connect to OSS for data upload and download, while only sending the control traffic to the app server.

Advantages

Setting up an OSS-based direct data transfer service for a mobile app offers the following advantages:

- More secure upload/download method (temporary and flexible permission assignment and authentication).
- Low cost. Fewer app servers. The mobile app is directly connected to the cloud storage and only the control traffic is sent to the app server.
- High concurrency and support for a massive amount of users (OSS has massive bandwidth for uploading and downloading use).
- Elasticity (OSS’s storage space can be expanded unlimitedly).
- Convenience. You can easily connect to the MTS -video multiport adapter, Image Service, CDN download acceleration, and other services.

The architecture diagram is as follows:



Details:

- Android/iOS mobile app, which is the app installed on the end user's mobile phone .
- OSS, short for Alibaba Cloud Object Storage Service, which stores app-uploaded data. For more information, see OSS description on Alibaba Cloud website.
- RAM/STS, which generates temporary access credentials.
- App server, which is the background service developed for the Android/iOS mobile app and used to manage the tokens used for data uploading/downloading by the app and the metadata of the app-uploaded data.

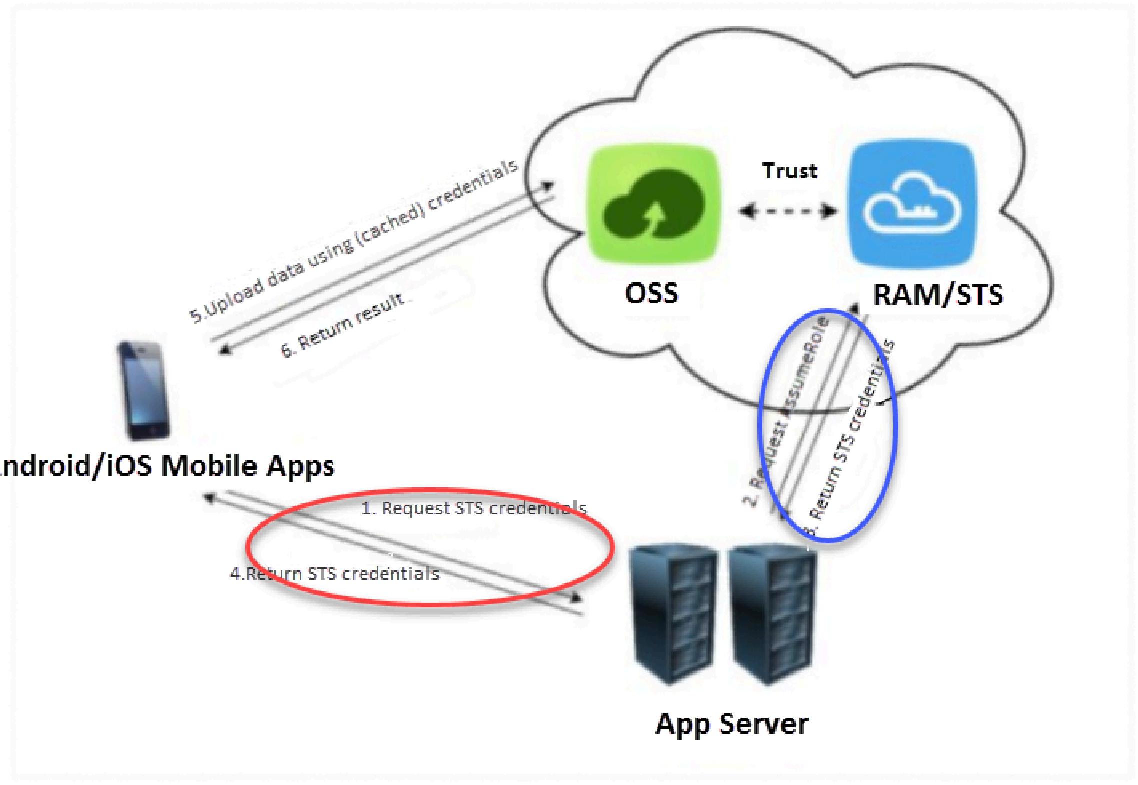
Steps:

1. Request for a temporary upload credential from the app server.

   The Android/iOS app cannot store AccessKeyID/AccessKeySecret directly, which may cause the risk of information leakage. Therefore, the app must request a temporary upload credential (a token) from the app server. The token is only valid for a certain period. For example, if a token is set to be valid for 30 minutes (editable by the app server), then the Android/iOS app can use this token to upload /download data to/from the OSS within the next 30 minutes. 30 minutes later, the app must request a new token to upload/download data.

2. The app server checks the validity of the preceding request and then returns a token to the app.
3. After the cell phone receives this token, it can upload or download data from the OSS.

This article mainly describes the content in the red circle and blue circle of the following figure.



- The blue circle shows how the app server generates a token.
- The red circle shows how the Android/iOS app receives the token.

Prerequisites for setting up direct data transfer service

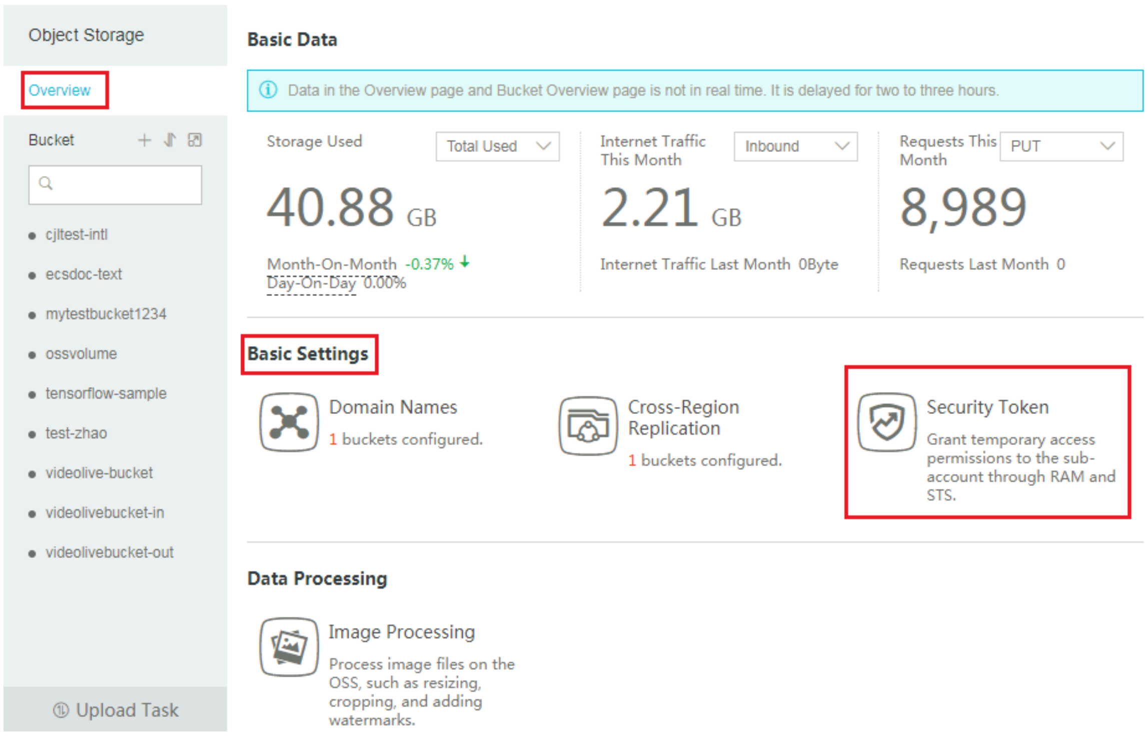
Preparations for setting up direct data transfer service:

1. Activate the OSS service and create a bucket.

2. Activate the STS service.

   a. Log on to the OSS console.

   b. On the OSS Overview page, find the `Basic Settings` area, and click Security Token, as shown in the following figure.

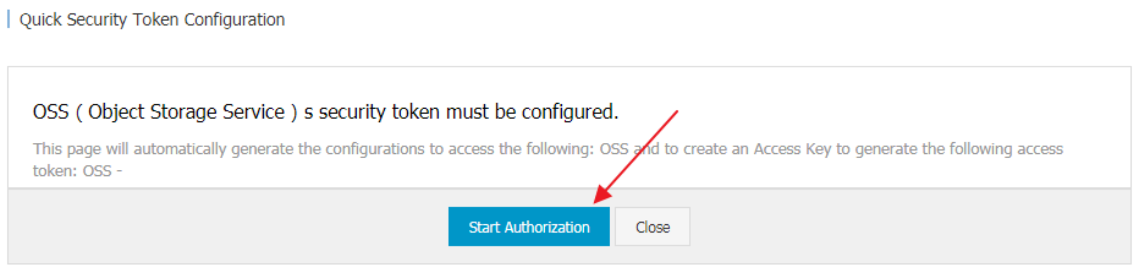


Data in the Overview page and Bucket Overview page is not in real time. It is delayed for two to three hours.

   c. Enter the Quick Security Token Configuration page.

Note:

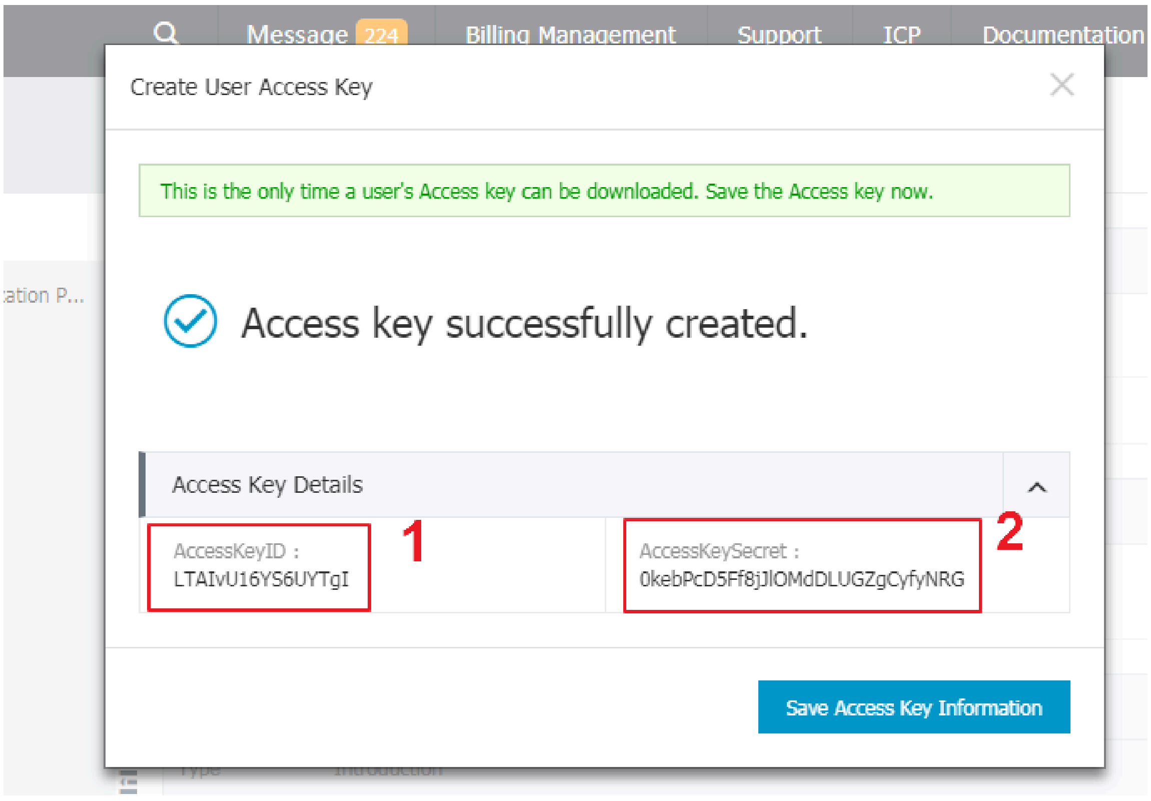
If RAM has not yet been activated, a prompt box to activate RAM appears. Click Activate and perform real-name verification. After the verification is finished, the following page appears. Click Start Authorization.


Quick Security Token Configuration

OSS (Object Storage Service) s security token must be configured.

This page will automatically generate the configurations to access the following: OSS and to create an Access Key to generate the following access token: OSS -

d. The system performs authorization automatically. Be sure to save the parameters in the three red boxesoxes in the following figures. Click Save Access Key Information to close the dialog box and complete STS activation.


Create User Access Key

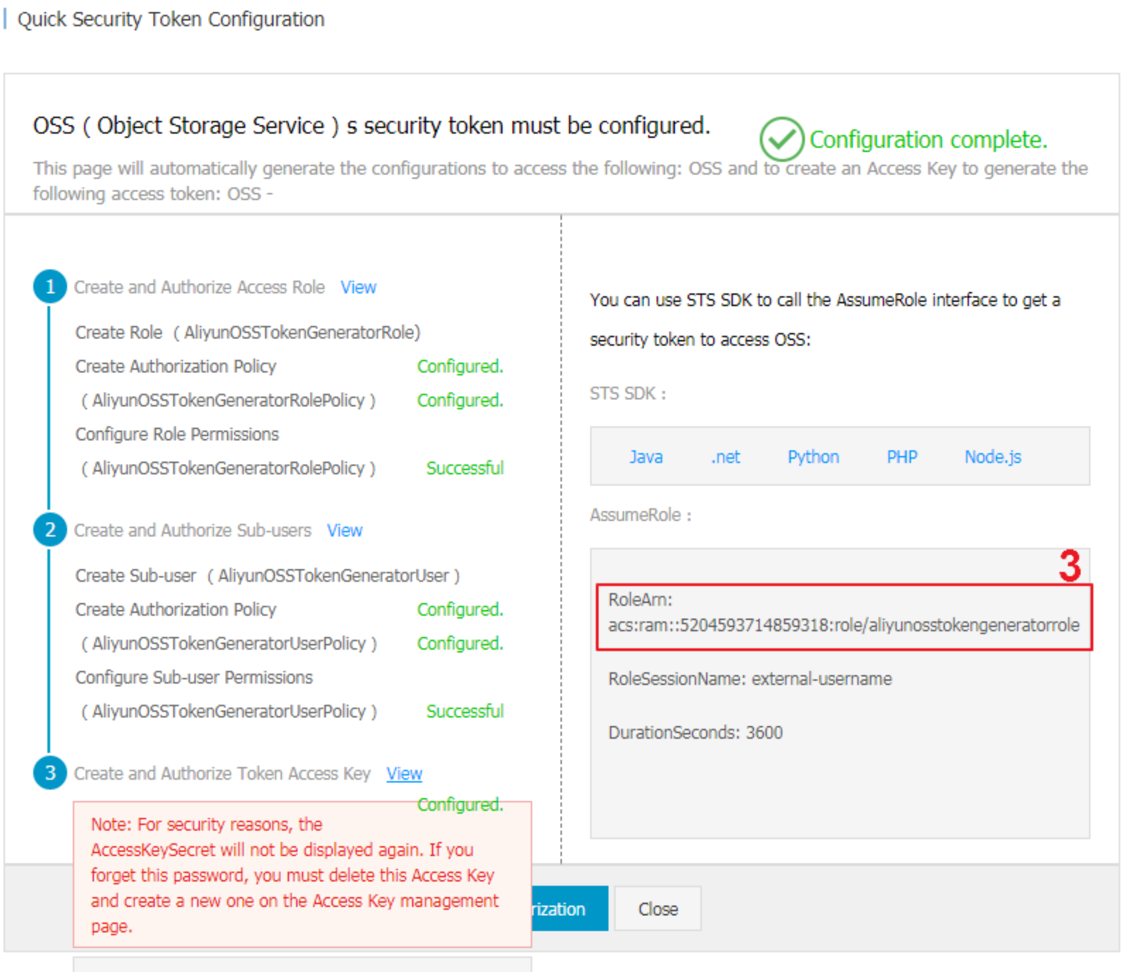
This is the only time a user's Access key can be downloaded. Save the Access key now.

Access key successfully created.

Access Key Details

AccessKeyID : LTAIvU16YS6UYTgI

AccessKeySecret : 0kebPcD5Ff8jJlOMdDLUGZgCyfyNRG



Quick Security Token Configuration

OSS (Object Storage Service) s security token must be configured.

Configuration complete.

This page will automatically generate the configurations to access the following: OSS and to create an Access Key to generate the following access token: OSS -

1 Create and Authorize Access Role View

Create Role (AliyunOSSTokenGeneratorRole)
Create Authorization Policy Configured.
(AliyunOSSTokenGeneratorRolePolicy) Configured.
Configure Role Permissions
(AliyunOSSTokenGeneratorRolePolicy) Successful

2 Create and Authorize Sub-users View

Create Sub-user (AliyunOSSTokenGeneratorUser)
Create Authorization Policy Configured.
(AliyunOSSTokenGeneratorUserPolicy) Configured.
Configure Sub-user Permissions
(AliyunOSSTokenGeneratorUserPolicy) Successful

3 Create and Authorize Token Access Key View

Configured.

Note: For security reasons, the AccessKeySecret will not be displayed again. If you forget this password, you must delete this Access Key and create a new one on the Access Key management page.

You can use STS SDK to call the AssumeRole interface to get a security token to access OSS:

STS SDK :

Java .net Python PHP Node.js

AssumeRole :

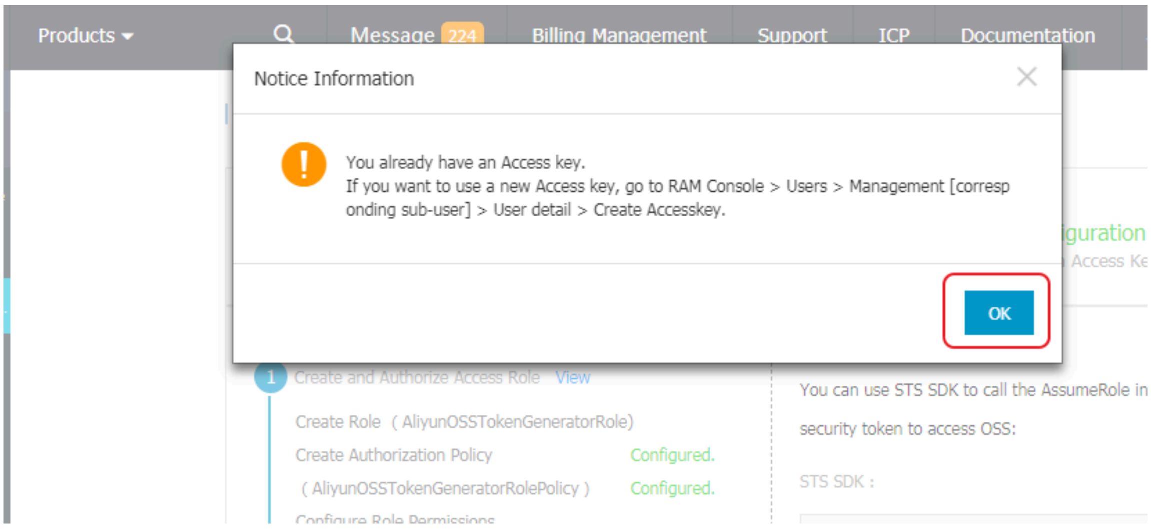
3

RoleArn:
acs:ram::5204593714859318:role/aliyunosstokengeneratorrole

RoleSessionName: external-username

DurationSeconds: 3600

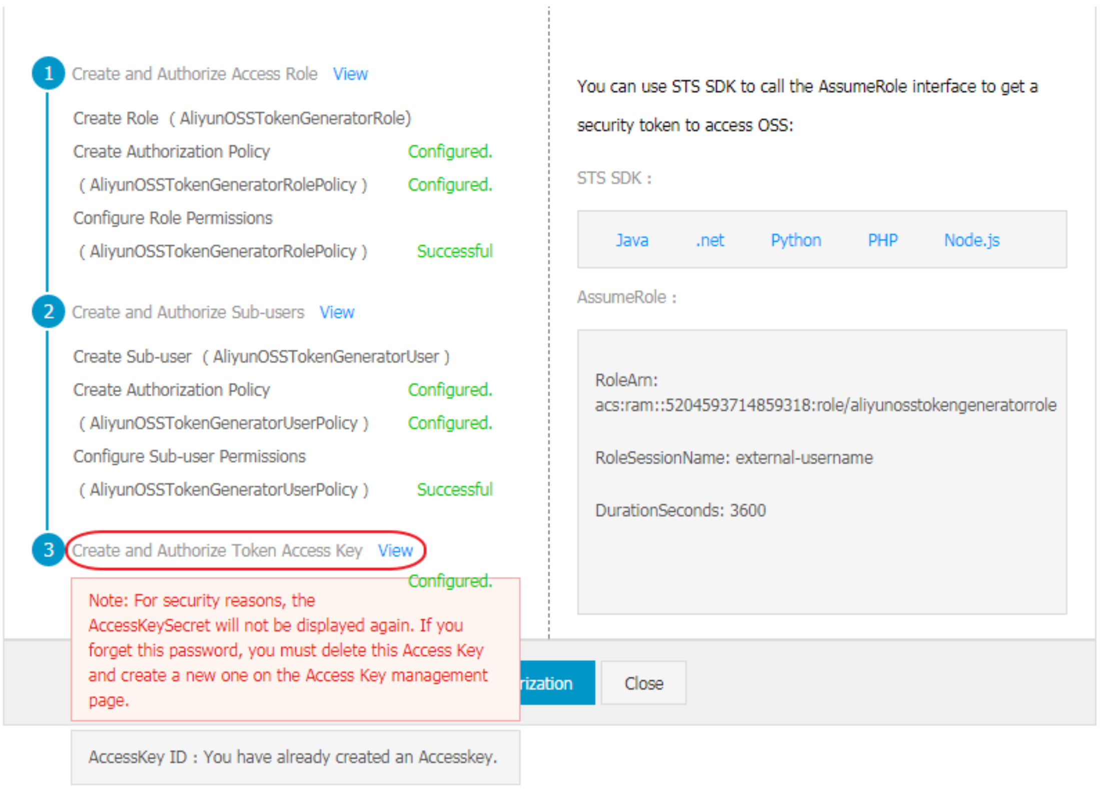
e. If you have already created an AccessKeyId/AccessKessKeySecret, the following prompt window appears:



Notice Information

You already have an Access key.
If you want to use a new Access key, go to RAM Console > Users > Management [corresponding sub-user] > User detail > Create Accesskey.

- Click View, as shown in the following figure.



1 Create and Authorize Access Role View

Create Role (AliyunOSSTokenGeneratorRole)
Create Authorization Policy Configured.
(AliyunOSSTokenGeneratorRolePolicy) Configured.
Configure Role Permissions
(AliyunOSSTokenGeneratorRolePolicy) Successful

2 Create and Authorize Sub-users View

Create Sub-user (AliyunOSSTokenGeneratorUser)
Create Authorization Policy Configured.
(AliyunOSSTokenGeneratorUserPolicy) Configured.
Configure Sub-user Permissions
(AliyunOSSTokenGeneratorUserPolicy) Successful

3 Create and Authorize Token Access Key View

Configured.

Note: For security reasons, the AccessKeySecret will not be displayed again. If you forget this password, you must delete this Access Key and create a new one on the Access Key management page.

AccessKey ID : You have already created an Accesskey.

You can use STS SDK to call the AssumeRole interface to get a security token to access OSS:

STS SDK :

Java .net Python PHP Node.js

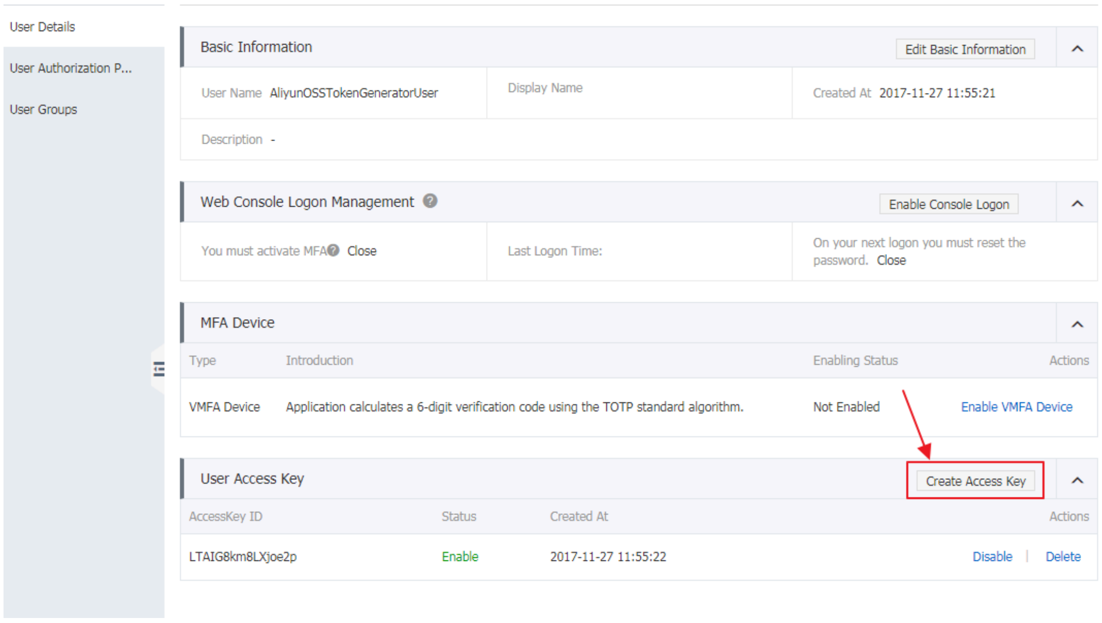
AssumeRole :

```
RoleArn:
acs:ram::5204593714859318:role/aliyunosstokengeneratorrole

RoleSessionName: external-username

DurationSeconds: 3600
```

- Click Create Access Key, as shown in the following figure.



Basic Information — Edit Basic Information

User Name AliyunOSSTokenGeneratorUser	Display Name	Created At 2017-11-27 11:55:21
Description -		

Web Console Logon Management — Enable Console Logon

You must activate MFA Close	Last Logon Time:	On your next logon you must reset the password. Close

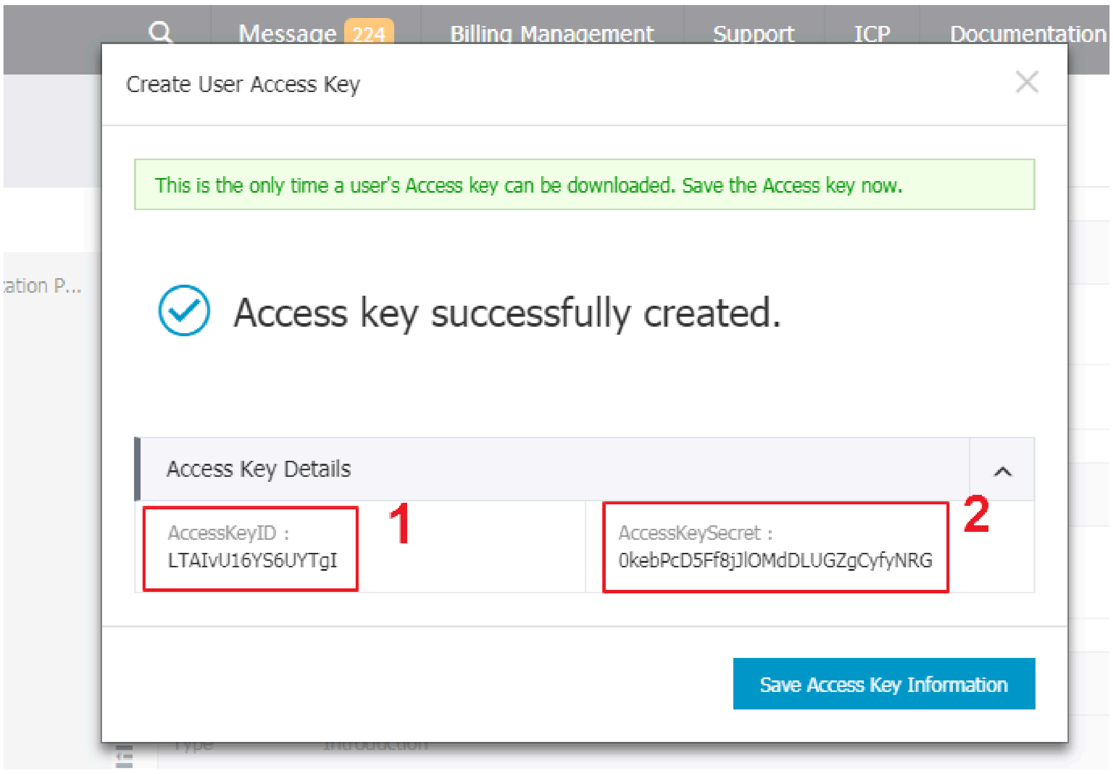
MFA Device

Type	Introduction	Enabling Status	Actions
VMFA Device	Application calculates a 6-digit verification code using the TOTP standard algorithm.	Not Enabled	Enable VMFA Device

User Access Key — Create Access Key

AccessKey ID	Status	Created At	Actions
LTAIG8km8LXjoe2p	Enable	2017-11-27 11:55:22	Disable | Delete

- Record parameters 1, 2, and 3, as shown in the following figure.



Create User Access Key

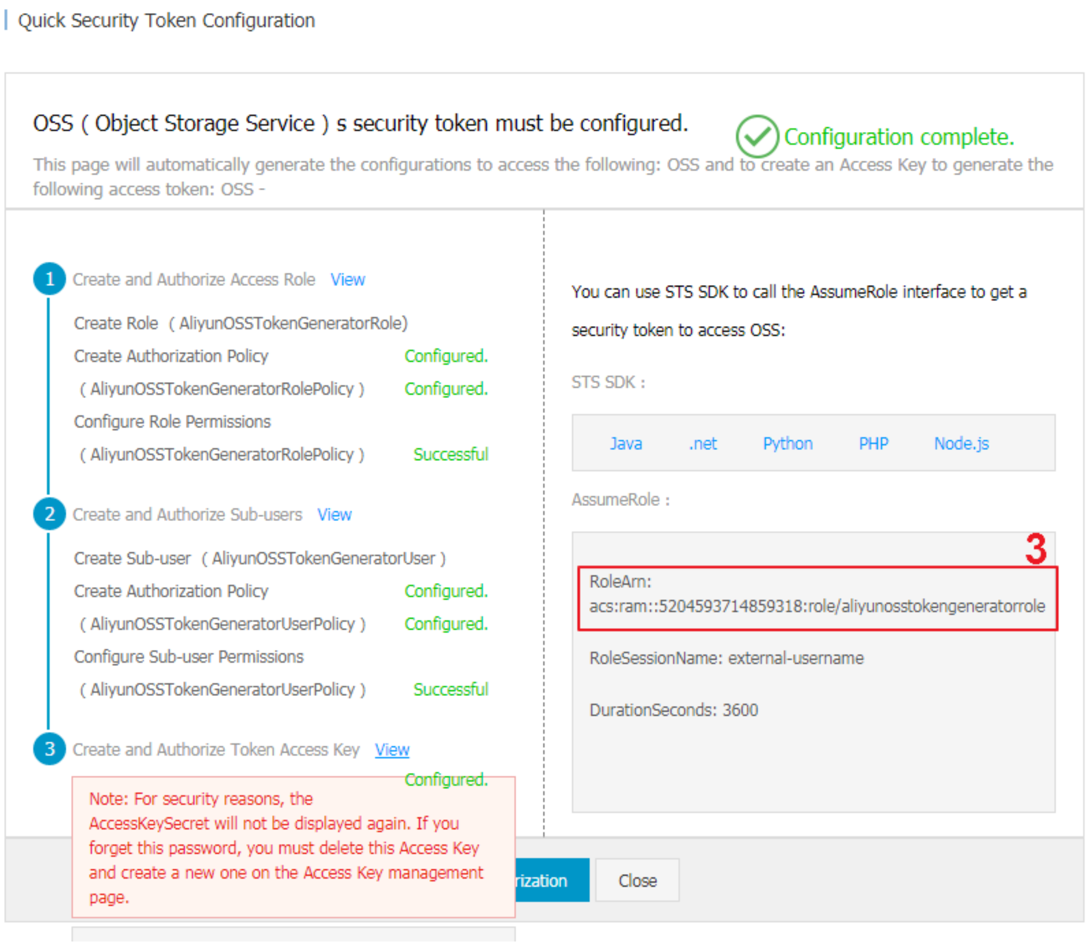
This is the only time a user's Access key can be downloaded. Save the Access key now.

Access key successfully created.

Access Key Details

AccessKeyID : LTAIvU16YS6UYTgI 1

AccessKeySecret : 0kebPcD5Ff8jJlOMdDLUGZgCyfyNRG 2


Quick Security Token Configuration

OSS (Object Storage Service) s security token must be configured. Configuration complete.

This page will automatically generate the configurations to access the following: OSS and to create an Access Key to generate the following access token: OSS -

1 Create and Authorize Access Role View

Create Role (AliyunOSSTokenGeneratorRole)
Create Authorization Policy Configured.
(AliyunOSSTokenGeneratorRolePolicy) Configured.
Configure Role Permissions
(AliyunOSSTokenGeneratorRolePolicy) Successful

2 Create and Authorize Sub-users View

Create Sub-user (AliyunOSSTokenGeneratorUser)
Create Authorization Policy Configured.
(AliyunOSSTokenGeneratorUserPolicy) Configured.
Configure Sub-user Permissions
(AliyunOSSTokenGeneratorUserPolicy) Successful

3 Create and Authorize Token Access Key View
Configured.

Note: For security reasons, the AccessKeySecret will not be displayed again. If you forget this password, you must delete this Access Key and create a new one on the Access Key management page.

You can use STS SDK to call the AssumeRole interface to get a security token to access OSS:

STS SDK :

AssumeRole :

3
RoleArn: acs:ram::5204593714859318:role/aliyunosstokengeneratorrole

RoleSessionName: external-username

DurationSeconds: 3600

- Once you have saved the three parameters, STS activation is complete.

Set up an app server

Configuration of sample app server

Note:

The app in this example is written in PHP. You may write your app in your preferred language, e.g. Java, Python, Go, Ruby, Node.js, or C#.

This tutorial provides development sample programs available for download in multiple languages. The download addresses are shown at the end of this article.

The downloaded package in each language contains a configuration file named config .json.

```
{
" AccessKeyI  D " : "",
" AccessKeyS  ecret " : "",
" RoleArn " : "",
" TokenExpir  eTime " : " 900 ",
" PolicyFile ": " policy / all_policy . txt "
}
```

Note:

- 1. AccessKeyID: Set it to parameter 1 marked with a red box in the preceding figure.
  2. AccessKeySecret: Set it to parameter 2 marked with a red box in the preceding figure.
  3. RoleArn: Set it to parameter 3 marked with a red box in the preceding figure.
  4. TokenExpireTime: indicates the expiration time of the token obtained by the Android/iOS app. The minimum value is 900s. The default value can be retained .
  5. PolicyFile: indicates the file that lists the permissions the token grants. The default value can be retained.

This document has provided three token files defining the most common permission s in the policy directory. They are:

- all_policy.txt: specifying a token that grants permissions to create or delete a bucket, or upload, download, or delete a file for this account .
- bucket_read_policy.txt: specifying a token that grants permission to read the specified bucket for this account.

- **bucket_read_write_policy.txt: specifying a token that grants permission to read and write the specified bucket for this account.**

If you want to create a token to grant read and write permissions for the specified bucket, replace $BUCKET_NAME in the bucket_read_policy.txt and bucket_read_write_policy.txt files with the name of the specified bucket.

- **Explanation of the formats of returned data:**

```
// Correct   result   returned
{
    " StatusCode ": 200 ,
    " AccessKeyI  d ":" STS .  3p *** dgagdasdg ",
    " AccessKeyS  ecret ":" rpnwO9 *** tGdrddgsR2  YrTtI ",
   " SecurityTo  ken ":" CAES + wMIARKAAZh  jH0EUOIhJM  QBMjRywXq7
 MQ / cjLYg80Aho  1ek0Jm63XM  hr9Oc5s ˙∂˙∂ 3qaPer8p1Y  aX1NTDiCFZ
 WFkvlHf1pQ  huxfKBc + mRR9KAbHUe  fqH + rdjZqjTF7p  2m1wJXP8S6
 k + G2MpHrUe6T  YBkJ43GhhT  VFMuM3BZaj  Y3VjZWOXBI  ODRIR1FKZj
 IiEjMzMzE0  MjY0NzM5MT  E4NjkxMSoL  Y2xpZGSSDg  SDGAGESGTE
 TqOio6c2Rr  LWRlbW8vKg  oUYWNzOm9z  czoqOio6c2  RrLWRlbW9K
 EDExNDg5Mz  AxMDcyNDY4  MThSBTI2OD  QyWg9Bc3N1  bWVkUm9sZV
 VzZXJgAGoS  MzMzMTQyNj  Q3MzkxMTg2  OTExcglzZG  stZGVtbzI =",
   " Expiration ":" 2015 - 12 - 12T07 : 49 : 09Z ",
}
// Wrong   result   returned
{
    " StatusCode ": 500 ,
    " ErrorCode ":" InvalidAcc  essKeyId . NotFound ",
    " ErrorMessa  ge ":" Specified   access   key   is   not   found
 ."
}
```

Explanation of correct result returned: (The following five variables comprise a token)

- StatusCode: The status indicates the result that the app retrieves the token. The app returns 200 for successful retrieval of the token.
- AccessKeyId: indicates the AccessKeyId the Android/iOS app obtains when initializing the OSS client.
- AccessKeySecret: indicates the AccessKeySecret the Android/iOS app obtains when initializing the OSS client.
- SecurityToken: indicates the token the Android/iOS app initializes.
- Expiration: indicates the time when the token expires. The Android SDK automatically determines the validity of the token and then retrieves a new one as needed.

Explanation of wrong result returned:

- StatusCode: The status indicates the result that the app retrieves the token. The app returns 500 for unsuccessful retrieval of the token.
- ErrorCode: indicates the error causes.
- ErrorMessage: indicates the detailed information about the error.

· Method for running sample code:

  - For PHP, download and unzip a pack, modify the config.json file, run php sts.php to generate a token, and deploy the program to the specified address.
  - For Java (based on Java 1.7), after downloading and unzipping a pack,

    Run this command `java - jar oss - token - server . jar ( port )`. If you run `java - jar oss - token - server . jar` without specifying a port, the program listens to Port 7080. To change the listening port to 9000, run `java - jar app - token - server . jar 9000` . Specify the port number as needed.

How to upload files from your app to oss

1. After setting up the app server, write down the server address, which is `http :// abc . com : 8080` . Then, replace the app server address in the sample project with this address.
2. Specify the bucket and region for the upload in the sample apps.

3. Click Set to load the configuration.
4. Select an image file, set the object name to upload to OSS, and select Upload. Now you can experience the OSS service on Android. Data from the Android app can be uploaded directly to OSS.
5. After the upload is complete, check that the data is on OSS.

Explanation of core code

OSS initialization

The following explains how to use the Android/iOS SDK to request a token from your app server.

- Android versions

```
// Initialize   an   OssService   for   upload   and   download .
 public   OssService   initOSS ( String   endpoint ,  String
 bucket ,  UIDisplaye  r   displayer ) {
     OSSCredent  ialProvide  r   credential  Provider ;
    // Use   your   own   class   to   retrieve   an   STSToken .
// Read   the   server   address   from   app   server   controls .
     String   stsServer   = (( EditText )  findViewBy  Id ( R . id .
 stsserver )). getText (). toString ();
    // STSGetter   class ,  encapsulat  ing   the   way   of
   retrieving   data   from   the   app   server ,  must   be
   inherited   from   the   class   OSSFederat  ionCredent
 ialProvide  r .  The   way   that   your   app   retrieves   tokens
   depends   on   the   protocol   between   the   app   and   the
 app   server .
     if  ( stsServer  . equals ("")) {
         credential  Provider  =  new   STSGetter ();
    } else  {
         credential  Provider  =  new   STSGetter ( stsServer );
    }
// Retrieve   the   bucket   name   from   the   controls .
     bucket  = (( EditText )  findViewBy  Id ( R . id . bucketname
 )). getText (). toString ();
// Initialize   an   OSSClient .
     ClientConf  iguration   conf  =  new   ClientConf  iguration
 ();
     conf . setConnect  ionTimeout ( 15  *  1000 ); //  Connection
 time - out .  The   default   value   is   15   seconds .
     conf . setSocketT  imeout ( 15  *  1000 ); //  Socket   time -
 out .  The   default   value   is   15   seconds .
     conf . setMaxConc  urrentRequ  est ( 5 ); //  The   maximum
 number   of   concurrent   requests .  The   default   value   is
 5 .
     conf . setMaxErro  rRetry ( 2 ); //  The   maximum   number
 of   retry   attempts   after   each   failed   attempt .  The
 default   value   is   2 .
     OSS   oss  =  new   OSSClient ( getApplica  tionContex  t (),
 endpoint ,  credential  Provider ,  conf );
     return   new   OssService ( oss ,  bucket ,  displayer );
}
```

- iOS version

```
// Initialize   an   OSSClient   instance .
- ( void ) ossInit  {
    // Construct   a   credential   provider   for   retrieving
 STSTokens .
     id < OSSCredent  ialProvide  r >  credential  = [[ OSSFederat
 ionCredent  ialProvide  r   alloc ]  initWithFe  derationTo
 kenGetter :^ OSSFederat  ionToken  * {
        // Implement   a   function   to   synchroniz  e   the
 STSToken   retrieved   from   the   server .
         return  [ self   getFederat  ionToken ];
    }];
    // Use   endpoint   and   the   credential   provider   to
 initialize   an   OSSClient .
     client  = [[ OSSClient   alloc ]  initWithEn  dpoint : endPoint
   credential  Provider : credential ];
}
```

Retrieve tokens from app server for mobile app

The specific method by which the app gets tokens from the app server must be written into the function `public   OSSFederat  ionToken   getFederat  ionToken () { }`.

Note:
you can define the logic for this function; however, the return message must contain this variable: *`return new OSSFederationToken(ak, sk, token, expiration)`*.Here, ak, sk, token, and expiration must be taken from the body of the message returned by the server.

In this example, you can specify the protocol linking the app and app server.

- Android version

```
 public   OSSFederat  ionToken   getFederat  ionToken () {
     String   stsJson ;
     OkHttpClie  nt   client  =  new   OkHttpClie  nt ();
     Request   request  =  new   Request . Builder (). url (
 stsServer ). build ();
     try  {
         Response   response  =  client . newCall ( request ).
 execute ();
         if  ( response . isSuccessf  ul ()) {
             stsJson  =  response . body (). string ();
        }  else  {
             throw   new   IOExceptio  n (" Unexpected   code  " +
 response );
        }
    }
     catch  ( IOExceptio  n   e ) {
         e . printStack  Trace ();
```

```
          Log . e (" GetSTSToke  nFail ",  e . toString ());
          return   null ;
      }
       try  {
          JSONObject   jsonObjs  =  new   JSONObject ( stsJson );
          String   ak  =  jsonObjs . getString (" AccessKeyI  d ");
          String   sk  =  jsonObjs . getString (" AccessKeyS  ecret
 ");
          String   token  =  jsonObjs . getString (" SecurityTo  ken
 ");
          String   expiration  =  jsonObjs . getString (" Expiration
 ");
          return   new   OSSFederat  ionToken ( ak ,  sk ,  token ,
 expiration );
      }
       catch  ( JSONExcept  ion   e ) {
          Log . e (" GetSTSToke  nFail ",  e . toString ());
          e . printStack  Trace ();
          return   null ;
      }}
```

· **iOS version**

```
 NSURL  *  url  = [ NSURL   URLWithStr  ing : STSServer ];
 NSURLReque  st  *  request  = [ NSURLReque  st   requestWit  hURL :
 url ];
 OSSTaskCom  pletionSou  rce  *  tcs  = [ OSSTaskCom  pletionSou
 rce   taskComple  tionSource ];
 NSURLSessi  on  *  session  = [ NSURLSessi  on   sharedSess  ion ];
 NSURLSessi  onTask  *  sessionTas  k  = [ session   dataTaskWi
 thRequest : request
                                            completion  Handler :^(
 NSData  * data ,  NSURLRespo  nse  * response ,  NSError  * error )
 {
                                                if  ( error ) {
                                                   [ tcs   setError
 : error ];
                                                    return ;
                                               }
                                               [ tcs   setResult :
 data ];
                                           }];
[ sessionTas  k   resume ];
//  Implementa  tion   of   this   callback   must   be
 synchroniz  ed   with   the   returned   token ,  so   the   task
 waitUntilF  inished   is   necessary .
[ tcs . task   waitUntilF  inished ];
 if  ( tcs . task . error ) {
    //  If   the   network   request   fails ,  the   return   of
   nil   indicates   the   token   cannot   be   retrieved .  In
 this   case ,  this   OSS   request   fails .
     return   nil ;
}  else  {
    //  Parse   the   JSON   string   returned   to   the   network
   request   to   get   each   token   field   and   return   an
 STSToken .
     NSDictiona  ry  *  object  = [ NSJSONSeri  alization
 JSONObject  WithData : tcs . task . result
                                                            options
 : kNilOption  s
                                                              error
 : nil ];
```

```
    OSSFederat  ionToken  *  token  = [ OSSFederat  ionToken   new
 ]; ni
    token . tAccessKey  = [ object   objectForK  ey :@" AccessKeyI
 d "];
    token . tSecretKey  = [ object   objectForK  ey :@" AccessKeyS
 ecret "];
    token . tToken  = [ object   objectForK  ey :@" SecurityTo  ken
 "];
    token . expiration  TimeInGMTF  ormat  = [ object   objectForK
 ey :@" Expiration "];
    return   token ;
}
```

Download source code

Example program

- Sample app source code for Android: download address
- Sample app source code for iOS: download address

Download sample code of app server

- PHP: download address
- Java: download address
- Ruby: download address
- node.js: download address

3.2 Permission control

This document elaborates how to configure different policies to implement different permission controls based on the app server mentioned in Set up direct data transfer for mobile apps by taking the app-base-oss bucket in the Shanghai region as an example.

Note:

- The following illustration assumes you have already activated STS and have thoroughly read the Set up direct data transfer for mobile apps document.
- The policies mentioned in the following content are covered in the specified policy file in the config.json file mentioned in the previous section.
- The operations on OSS upon retrieving the STS token indicate the process of specifying the policy for the app server, the app server retrieving a temporary credential from the STS and the app using the temporary credential to access OSS.

Common policies

- Full authorization policy

  For the ease of demonstration, the default policy is shown as follows. This policy indicates that the app is allowed to perform any operation on OSS.

  Note:
  This policy is neither secured nor recommended to use for mobile apps.

```
{
    " Statement ": [
      {
        " Action ": [
          " oss :*"
        ],
        " Effect ": " Allow ",
        " Resource ": [" acs : oss :*:*:*"]
      }
    ],
    " Version ": " 1 "
  }
```

Operations on OSS upon retrieving STS token	Result
List all created buckets.	Successful
Upload the object without a prefix, test. txt.	Successful
Download the object without a prefix, test.txt.	Successful
Upload the object with a prefix, user1/ test.txt.	Successful
Download the object with a prefix, user1/test.txt.	Successful
List the object without a prefix, test.txt.	Successful
List the object with a prefix, user1/test. txt.	Successful

- Read-only policies with or without any prefixes

  This policy indicates the app can list and download all objects in the bucket app-base-oss.

```
{
      " Statement ": [
        {
          " Action ": [
            " oss : GetObject ",
            " oss : ListObject  s "
          ],
          " Effect ": " Allow ",
          " Resource ": [" acs : oss :*:*: app - base - oss /*", "
 acs : oss :*:*: app - base - oss "]
        }
      ],
      " Version ": " 1 "
    }
```

Operations on OSS upon retrieving STS token	Result
List all created buckets.	Failed
Upload the object without a prefix, test.txt.	Failed
Download the object without a prefix, test.txt.	Successful
Upload the object with a prefix, user1/test.txt.	Failed
Download the object with a prefix, user1/test.txt.	Successful
List the object without a prefix, test.txt.	Successful
List the object with a prefix, user1/test.txt.	Successful

- Read-only policies with a specified prefix

  This policy indicates the app can list and download all objects with the prefix of ** user1/** in the bucket **app-base-oss**. However, the policy does not specify to download any objects with another prefix. By this way, different apps correspond ing to different prefixes are spatially isolated in the bucket.

```
{
    " Statement ": [
      {
        " Action ": [
          " oss : GetObject ",
          " oss : ListObject  s "
        ],
        " Effect ": " Allow ",
        " Resource ": [" acs : oss :*:*: app - base - oss / user1 /
*", " acs : oss :*:*: app - base - oss "]
      }
    ],
    " Version ": " 1 "
```

```
  }
```

Operations on OSS upon retrieving STS token	Result
List all created buckets.	Failed
Upload the object without a prefix, test.txt.	Failed
Download the object without a prefix, test.txt.	Failed
Upload the object with a prefix, user1/test.txt.	Failed
Download the object with a prefix, user1/test.txt.	Successful
List the object without a prefix, test.txt.	Successful
List the object with a prefix, user1/test.txt.	Successful

- Write-only policies with no specified prefixes

  This policy indicates that the app can upload all objects in the bucket app-base-oss.

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

Operations on OSS upon retrieving STS token	Result
List all created buckets.	Failed
Upload the object without a prefix, test.txt.	Successful
Download the object without a prefix, test.txt.	Failed
Upload the object with a prefix, user1/test.txt.	Successful

Operations on OSS upon retrieving STS token	Result
Download the object with a prefix, user1/test.txt.	Successful
List the object without a prefix, test.txt.	Successful
List the object with a prefix, user1/test.txt.	Successful

- Write-only policies with a specified prefix

  This policy indicates the app can upload all objects with the user1/ prefix in the bucket app-base-oss. The app cannot upload any object with another prefix. In this way, different apps corresponding to different prefixes are spatially isolated in the bucket.

```
{
    " Statement ": [
      {
        " Action ": [
          " oss : PutObject "
        ],
        " Effect ": " Allow ",
        " Resource ": [" acs : oss :*:*: app - base - oss / user1 /
*", " acs : oss :*:*: app - base - oss "]
      }
    ],
    " Version ": " 1 "
  }
```

Operations on OSS upon retrieving STS token	Result
List all created buckets.	Failed
Upload the object without a prefix, test.txt.	Failed
Download the object without a prefix, test.txt.	Failed
Upload the object with a prefix, user1/test.txt.	Successful
Download the object with a prefix, user1/test.txt.	Failed
List the object without a prefix, test.txt.	Failed
List the object with a prefix, user1/test.txt.	Failed

- Read/write policies with or without any prefixes

  This policy indicates that the app can list, download, upload, and delete all objects in the bucket `app - base - oss` .

```
{
    " Statement ": [
      {
        " Action ": [
          " oss : GetObject ",
          " oss : PutObject ",
          " oss : DeleteObje  ct ",
          " oss : ListParts ",
          " oss : AbortMulti  partUpload ",
          " oss : ListObject  s "
        ],
        " Effect ": " Allow ",
        " Resource ": [" acs : oss :*:*: app - base - oss /*", " acs
 : oss :*:*: app - base - oss "]
      }
    ],
    " Version ": " 1 "
  }
```

Operations on OSS upon retrieving STS token	Result
List all created buckets.	Failed
Upload the object without a prefix, test.txt.	Successful
Download the object without a prefix, test.txt.	Successful
Upload the object with a prefix, user1/test.txt.	Successful
Download the object with a prefix, user1/test.txt.	Successful
List the object without a prefix, test.txt.	Successful
List the object with a prefix, user1/test.txt.	Successful

- Read/write policies with a specified prefix

  This policy indicates the app can list, download, upload, and delete all objects with a prefix of `user1 /` in the bucket `app - base - oss` . The policy does not specify to read or write any objects with another prefix. In this way, different apps corresponding to different prefixes are spatially isolated in the bucket.

```
{
    " Statement ": [
      {
        " Action ": [
          " oss : GetObject ",
          " oss : PutObject ",
          " oss : DeleteObje  ct ",
          " oss : ListParts ",
          " oss : AbortMulti  partUpload ",
          " oss : ListObject  s "
        ],
        " Effect ": " Allow ",
        " Resource ": [" acs : oss :*:*: app - base - oss / user1 /
*", " acs : oss :*:*: app - base - oss "]
      }
    ],
    " Version ": " 1 "
  }
```

Operations on OSS upon retrieving STS token	Result
List all created buckets.	Failed
Upload the object without a prefix, test.txt.	Failed
Download the object without a prefix, test.txt.	Failed
Upload the object with a prefix, user1/test.txt.	Successful
Download the object with a prefix, user1/test.txt.	Successful
List the object without a prefix, test.txt.	Successful
List the object with a prefix, user1/test.txt.	Successful

Summary

With the help of preceding examples, we can understand that:

- You can create different policies for various app scenarios and then achieve differentiated permission control for different apps through slight modifications on the app server.
- You can also optimize apps to save the process of making another request to the app server before the STS token expires.
- Tokens are actually issued by the STS. An app server customizes a policy, requests for a token from the STS, and then delivers this token to the app. Here, token is only a shorthand expression. However, a "token" actually contains an "AccessKeyI

d", an "AccessKeySecret", an "Expiration" value, and a "SecurityToken". These are used in the SDK provided by OSS to the app. For more information, see the implementation of the respective SDK.

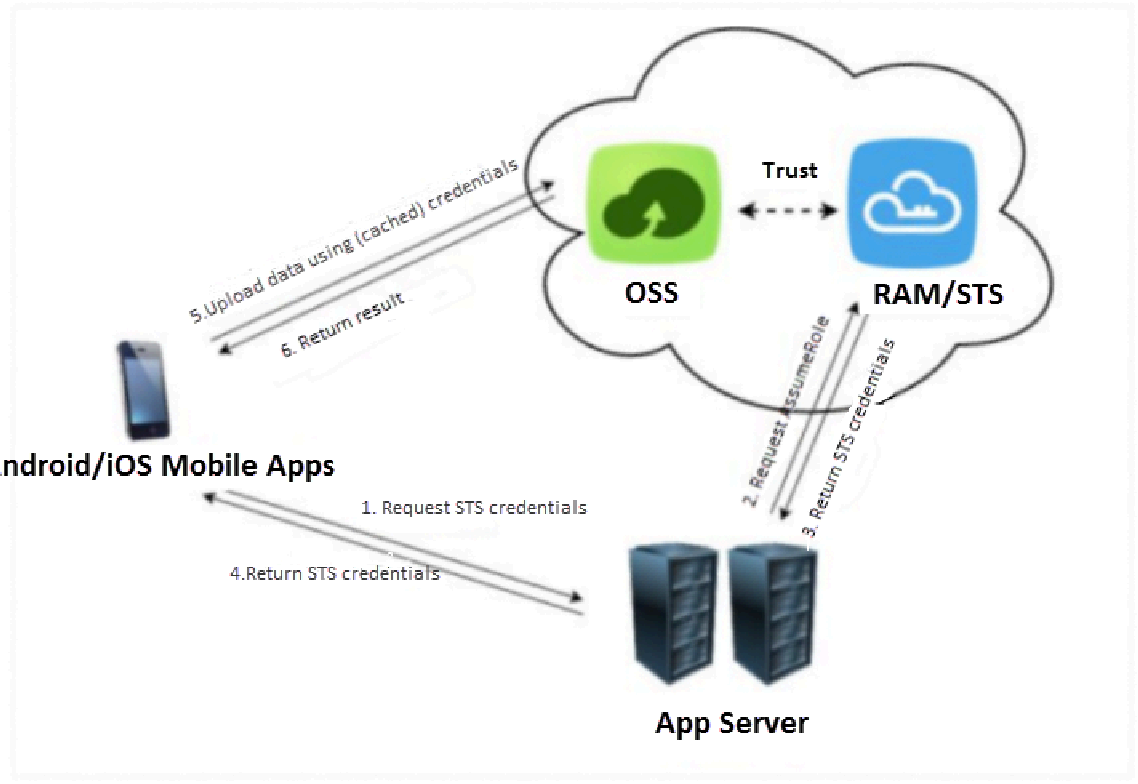
More references:

- How to use RAM and STS in OSS
- RAM documentation and STS documentation

3.3 Set up data callback for mobile apps

Background

Setting up direct data transfer for mobile apps describes how to set up OSS-based direct data transfer for mobile apps in 30 minutes. The following flowchart describes mobile app development:

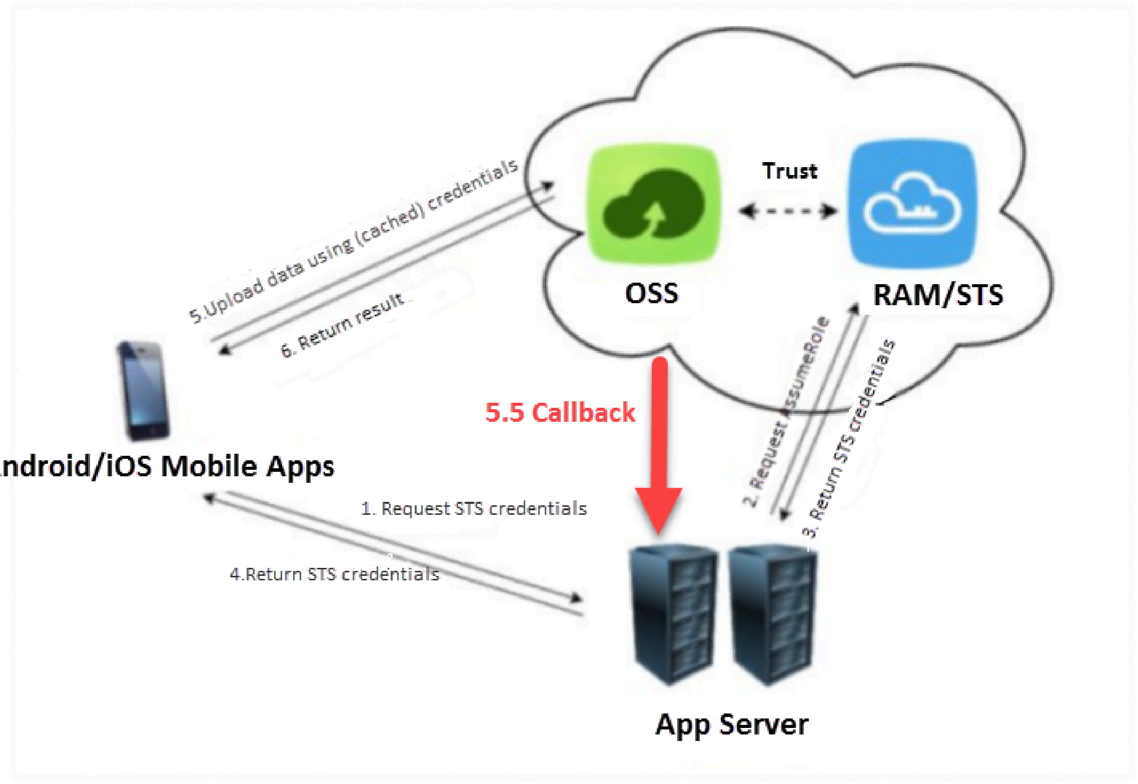


Role:

- The app server generates an STS credential for the Android/iOS mobile app.
- The Android/iOS mobile app applies for the STS credential from the app server and then uses the STS credential.
- The OSS processes requests from the Android/iOS mobile app.

After performing Step 1 (apply for an STS credential) in the preceding flowchart, the Android/iOS mobile app, can perform Step 5 (use the STS credential to upload data to OSS) repeatedly. In this case, the app server does not know what data the app is uploading, and the app developer cannot manage the uploaded data. Is there any way to make the app server be aware of the data uploaded by the Android/iOS mobile app?

In this case, the OSS data callback service can be used to tackle these type of issues. You can see the following flowchart:



OSS triggers a callback after receiving data from the Android/iOS mobile app (Step 5 in the preceding flowchart) but before returning the upload result to the app (Step 6). The callback is marked as Step 5.5. OSS calls back data from the app server and obtains the content returned by the app server. Then OSS returns the content to the Android/iOS mobile app. For more information, see Callback API Documentation.

Data callback function

- Retrieving basic information about the data uploaded to the app server

  The following table shows the basic information. One or more of the following variables are returned, and the format of returned content is specified when the Android/iOS mobile app uploads data.

System variable	Meaning
bucket	Storage space (bucket) to which the mobile app uploads data
object	File name saved on OSS for the data uploaded by the mobile app
etag	etag of the uploaded file. It is the etag field returned to the mobile app
size	Size of the uploaded file
mimeType	Resource type
imageInfo.height	Image height
imageInfo.width	Image width
imageInfo.format	Image format, for example, JPG and PNG (only for recognized images)

- Transferring information through custom variables

  If you are a developer and want to know the app version, OS version, location, and mobile phone model of the user who is uploading data, you can specify the Android/iOS mobile app client to send the preceding variables when uploading files. For example,

  - x:version indicates app version.
  - x:system indicates OS version.
  - x:gps indicates location.
  - x:phone indicates mobile phone model.

    These values are attached when the Android/iOS mobile app uploads data to OSS . Then OSS includes the values in the CallbackBody and sends them to the app server. In this way, the information is transferred to the app server.

Data callback setup for the mobile app client

To enable OSS to trigger a callback when receiving an upload request, the mobile app must include the following two items in the request:

- callbackUrl indicates the app server to which data is called back, for example, `http :// abc . com / callback . php` . Note that the server address must be accessible through the Internet.
- callbackBody indicates the content to be called back and sent to the app server. The content can include one or more of the variables OSS returns to the app server.

For example, assume that the data is called back and sent to the app server at `http :// abc . com / callback . php` . You want to obtain the name and size of the file uploaded by the mobile phone. The defined variable "photo" gets the mobile phone model, and the variable "system" gets the OS version.

Two samples of upload callbacks are listed as follows:

- Data callback sample code for iOS apps:

```
 OSSPutObje  ctRequest  *  request  = [ OSSPutObje  ctRequest   new
 ];
 request . bucketName  = @"< bucketName >";
 request . objectKey  = @"< objectKey >";
 request . uploadingF  ileURL  = [ NSURL   fileURLWit  hPath :@<
 filepath >"];
//  Set   callback   parameters
 request . callbackPa  ram  = @{
                          @" callbackUr  l ": @" http :// abc . com
 / callback . php ",
                          @" callbackBo  dy ": @" filename =${
 object }& size =${ size }& photo =${ x : photo }& system =${ x :
 system }"
                          };
//  Set   custom   variables
 request . callbackVa  r  = @{
                        @" x : photo ": @" iphone6s ",
                        @" x : system ": @" ios9 . 1 "
                        };
```

- Data callback sample code for Android apps:

```
 PutObjectR  equest   put  =  new   PutObjectR  equest ( testBucket
 ,  testObject ,  uploadFile  Path );
 ObjectMeta  data   metadata  =  new   ObjectMeta  data ();
 metadata . setContent  Type (" applicatio  n / octet - stream ");
 put . setMetadat  a ( metadata );
 put . setCallbac  kParam ( new   HashMap < String ,  String >() {
    {
        put (" callbackUr  l ", " http :// abc . com / callback .
 php ");
        put (" callbackBo  dy ", " filename =${ object }& size =${
 size }& photo =${ x : photo }& system =${ x : system }");
    }
```

```
});
 put . setCallbac  kVars ( new   HashMap < String ,  String >() {
     {
          put (" x : photo ", " IPOHE6S ");
          put (" x : system ", " YunOS5 . 0 ");
     }
});
```

Data callback requirements for the app server

- You must deploy a service for receiving POST requests. This service must have a public address, for example, `www . abc . com / callback . php` (or an Internet IP address); otherwise, OSS cannot access this address.
- You must set the format of custom content returned to OSS to JSON. OSS delivers the content received from the app server as it is to the Android/iOS mobile app. (The Response header returned to OSS must carry the Content-Length header.)

The last section provides sample callback programs based on multiple programming languages, together with the download links and running methods.

Callback request received by the app server

The packet of a callback request the app server receives from OSS is as follows (the data varies with different URLs and callback content):

```
POST  / index . html   HTTP / 1 . 0
Host :  121 . 43 . 113 . 8
Connection :  close
Content - Length :  81
Content - Type :  applicatio  n / x - www - form - urlencoded
User - Agent :  ehttp - client / 0 . 0 . 1
authorizat  ion :  kKQeGTRccD  KyHB3H9vF + xYMSrmhMZj  zzl2 /
kdD1ktNVgb  WEfYTQG0G2  SU / RaHBovRCE8  OkQDjC3uG3  3esH2txA ==
x - oss - pub - key - url :  aHR0cDovL2  dvc3NwdWJs  aWMuYWxpY2
RuLmNvbS9j  YWxsYmFja1  9wdWJfa2V5  X3YxLnBlbQ ==
filename = test . txt & size = 5 & photo = iphone6s & system = ios9 .
1
```

For more information, see Callback API Documentation.

How does the app server determine whether a callback request is sent by OSS?

The app server must determine whether a callback request is from OSS because the app server may receive invalid requests that affect its normal logic when the app server has a malicious callback during a network attack.

To determine request validity, the app server verifies the RSA checksum using the `x - oss - pub - key - url` and `authorizat  ion` parameters in the content OSS sends to the app server. Only requests that pass RSA checksum verification are sent by OSS. The sample programs in this document also provides implementation results, for your reference.

How does the app server process the received callback request?

After verifying a request from OSS, the app server processes the request based on its content. The Android/iOS mobile app specifies the format of the callback content when uploading the data, for example:

```
filename = test . txt & size = 5 & photo = iphone6s & system = ios9 .
1
```

The app server parses the OSS-returned content to obtain the expected data. Then the app server stores the data for subsequent management.

How does the app server return the callback request to OSS?

- The returned status code is 200.
- The returned content must use the JSON format.
- The returned content must carry the Content-Length header.

How does OSS process the content returned by the app server?

There are two scenarios:

- In case that the app server fails to receive the callback request or is not accessible , OSS returns a 203 status code to the Android/iOS mobile app. However, the uploaded data is already saved to OSS.
- If the app server receives a callback request and returns the correct status code, OSS returns content received from the app server as it is to the Android/iOS mobile app along with a 200 status code.

Sample callback programs for downloading

The sample program shows how to check the signature received by the application server. You must add the code for parsing the format of the callback content received by the application server.

- Java version:
  - Download address.
  - Running method: Extract the archive and run `java -jar oss-callback-server-demo.jar 9000` (9000 is the port number and can be changed as required).

    > Note:
    > This jar runs on java 1.7. If any problem occurs, you may make changes based on the provided code. This is a maven project.
- PHP version:
  - Download address.
  - Running method: Deploy the program to an Apache environment. Due to the characteristics of the PHP language, retrieving headers depends on the environment. You may make modifications to the example based on your own environment.
- Python version:
  - Download address.
  - Running method: Extract the archive and directly run python callback_a pp_server.py. The program implements a simple HTTP server. To run this program, you may need to install the system environment on which the RSA depends.
- Ruby version:
  - Download address.
  - Running method: ruby aliyun_oss_callback_server.rb

4 Data processing and analysis

4.1 Construct a data warehouse by using OSS and MaxCompute

This topic describes the method of using MaxCompute to construct a PB data warehouse based on OSS. By using MaxCompute to analyze the massive data stored in OSS, you can complete your big data analysis tasks in minutes and explore data value more efficiently.

Features

- Object Storage Service (OSS)

  OSS provides three storage classes: Standard, Infrequent Access, and Archive, which are suitable for different data access scenarios. OSS can be used together with Hadoop open-source community products and multiple Alibaba Cloud products, such as EMR, BatchCompute, MaxCompute, machine learning tool PAI, Data Lake Analytics, and Function Compute.

  You can create several data analysis applications by using OSS, including:

  - Batch processing applications using MapReduce, HIVE, Pig, or Spark, such as offline log computing
  - Interactive query analysis applications, such as Imapla, Presto, and Data Lake Analytics
  - Deep machine training applications, such as Alibaba Cloud PAI
  - Gene rendering computing and delivery applications, such as BatchCompute
  - Big data applications, such as MaxCompute
  - Flow processing applications, such as Function Compute
- MaxCompute

  MaxCompute is a big data computing service that provides fast and fully managed data warehouse solutions. MaxCompute can be used together with OSS to analyze and process massive data efficiently. With world-leading processing performanc e, MaxCompute was rated as the world's leading cloud-based data warehouse by Forrester.

- OSS-external table query

  One major feature of MaxCompute is the OSS-external table query function, which can help you query massive objects stored in OSS directly without needing to load data into MaxCompute tables. This can help you move data more efficiently and eliminates the need to store data in multiple places.

A data warehouse solution constructed by using MaxCompute and OSS has the following advantages:

- MaxCompute operates on a serverless, distributed computing architecture. Therefore, it allows for temporary query services in a timely manner based on the requirements of OSS users and does not require additional maintenance or management for server infrastructures, significantly reducing enterprise costs.
- OSS provides storage for massive data that can be accessed by multiple computing applications and services. As a result, you can effectively separate computing and storage resources by storing only one copy of data in OSS.
- A data warehouse solution using OSS and MaxCompute allows you to easily process structured files in open-source formats. Currently, the open-source formats supported are Avro, CSV, ORC, Parquet, RCFile, RegexSerDe, SequenceFile, and TextFile. The gzip compression format is also supported.

Application scenario

Financial applications on the Internet need to store a large number of financial data exchange files on OSS every day and must perform structured analysis on large test files. By using the OSS-external table query feature of MaxCompute, these requirements can be met with greater ease. Large files can be loaded on OSS to MaxCompute by using external tables. This process can significantly improve overall efficiency.

Example: Analyze data collected from Internet of Things (IoT)

1. Activate OSS. For more information, see #unique_42.
2. Create a bucket. For more information, see Create a bucket.
3. Activate MaxCompute. For more information, see #unique_51.
4. Create a MaxCompute project. For more information, see Create a project.

5. Grant OSS access permissions to MaxCompute.

   You must grant OSS access permissions to the account you used to access MaxCompute because MaxCompute needs to directly access data in OSS. You can log on to the RAM console with your Alibaba Cloud account to grant permissions.
6. Upload the data collected from IoT to OSS.

   Note:
   You can use any data set to test the procedures described in this topic. In this example, a CSV file named *vehicle . csv* has been uploaded to the / *demo* directory in an OSS bucket named oss-odps-test. The endpoint of the bucket is oss-cn-beijing-internal.aliyuncs.com.
7. Run the following commands to create an external table in MaxCompute. For more information, see Create a table.

```
 CREATE   EXTERNAL   TABLE   IF   NOT   EXISTS   ambulance_
 data_csv_e  xternal
(
     vehicleId   int ,
     recordId   int ,
     patientId   int ,
     calls   int ,
     locationLa  titute   double ,
     locationLo  ngtitue   double ,
     recordTime   string ,
     direction   string
    )
     STORED   BY  ' com . aliyun . odps . CsvStorage  Handler '
     LOCATION  ' oss :// oss - cn - beijing - internal . aliyuncs .
 com / oss - odps - test / Demo /';
```

8. Use MaxCompute to query the external table. You can use an external table in the same way as you use a normal table. For more information, see #unique_54/unique_54_Connect_42_section_ynz_3mq_kgb.

   Assume that the / *demo* / *vehicle . csv* file includes the following data:

```
 1 , 1 , 51 , 1 , 46 . 81006 ,- 92 . 08174 , 9 / 14 / 2014   0 : 00
 , S
 1 , 2 , 13 , 1 , 46 . 81006 ,- 92 . 08174 , 9 / 14 / 2014   0 : 00
 , NE
 1 , 3 , 48 , 1 , 46 . 81006 ,- 92 . 08174 , 9 / 14 / 2014   0 : 00
 , NE
 1 , 4 , 30 , 1 , 46 . 81006 ,- 92 . 08174 , 9 / 14 / 2014   0 : 00
 , W
 1 , 5 , 47 , 1 , 46 . 81006 ,- 92 . 08174 , 9 / 14 / 2014   0 : 00
 , S
 1 , 6 , 9 , 1 , 46 . 81006 ,- 92 . 08174 , 9 / 14 / 2014   0 : 00 ,
 S
 1 , 7 , 53 , 1 , 46 . 81006 ,- 92 . 08174 , 9 / 14 / 2014   0 : 00
 , N
```

```
1 , 8 , 63 , 1 , 46 . 81006 ,- 92 . 08174 , 9 / 14 / 2014   0 : 00
, SW
1 , 9 , 4 , 1 , 46 . 81006 ,- 92 . 08174 , 9 / 14 / 2014   0 : 00 ,
NE
1 , 10 , 31 , 1 , 46 . 81006 ,- 92 . 08174 , 9 / 14 / 2014   0 : 00
, N
```

Run the following SQL statement:

```
select   recordId ,  patientId ,  direction   from   ambulance_
data_csv_e  xternal   where   patientId  >  25 ;
```

The following results are returned.

```
+------------+------------+-----------+
|  recordId    |  patientId    |  direction  |
+------------+------------+-----------+
|  1           |  51           |  S            |
|  3           |  48           |  NE           |
|  4           |  30           |  W            |
|  5           |  47           |  S            |
|  7           |  53           |  N            |
|  8           |  63           |  SW           |
|  10          |  31           |  N            |
+------------+------------+-----------+
```

For more information about the usage of OSS-external tables, see #unique_55.

4.2 EMR+OSS: Separated storage and computing for offline computing

Background

In traditional Hadoop usage, storage and computing are inseparable. Therefore, as your business grows, the cluster size often cannot meet your needs for business expansion. For example, if your data scale exceeds the cluster' s storage capacity, the new requirements arising from the data production cycle of your business may outpace computing capabilities. In this case, you must be ready at all times to deal with the challenges of insufficient cluster storage space or computing capabilities.

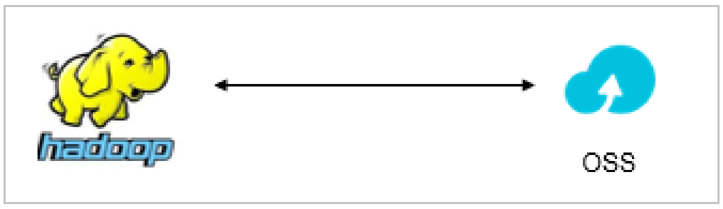
If you choose to deploy computing and storage in a hybrid manner, storage scaling can often lead to excess computing capabilities. This is a waste of resources. Likewise , an increase in computing capabilities causes a waste of storage resources.

Separating computing and storage for offline computing makes it easier to cope with insufficient computing or storage resources. In this solution, you can store all your data in OSS and then analyze it using stateless E-MapReduce. Therefore, E-

MapReduce is only responsible for computation, and storage resource are not tied to computing resources in your business. This approach provides the highest flexibility.

Architecture

The architecture for offline computing with separated storage and computing is simple, as shown in the following figure. OSS acts as the default storage unit, and Hadoop or Spark acts as a computing engine that directly analyzes data stored in OSS.



Benefits

Factor	Integrated computing and storage	Separated computing and storage
Flexibility	Not flexible	After computing and storage are separated, cluster rules are simple and flexible. You hardly need to estimate your future business scale , besides using the resources as you need.
Cost	High	Ultra cloud disks are used in self-built ECS systems . After separating storage and computing, if the cluster configuration is one master node with an 8-core 32 GB CPU, six slave nodes with 8-core 32 GB CPUs, and 10 TB of data, the cost is roughly halved.
Performance	Relatively high	At most, performance drops by 10%.

Test case

- **Test conditions**

  For the detailed test code, see GitHub.

  Cluster scale: 1 master node with a 4-core 16 GB CPU, 8 slave nodes with 4-core 16 GB CPUs, each slave node has four 250 GB ultra cloud disks.

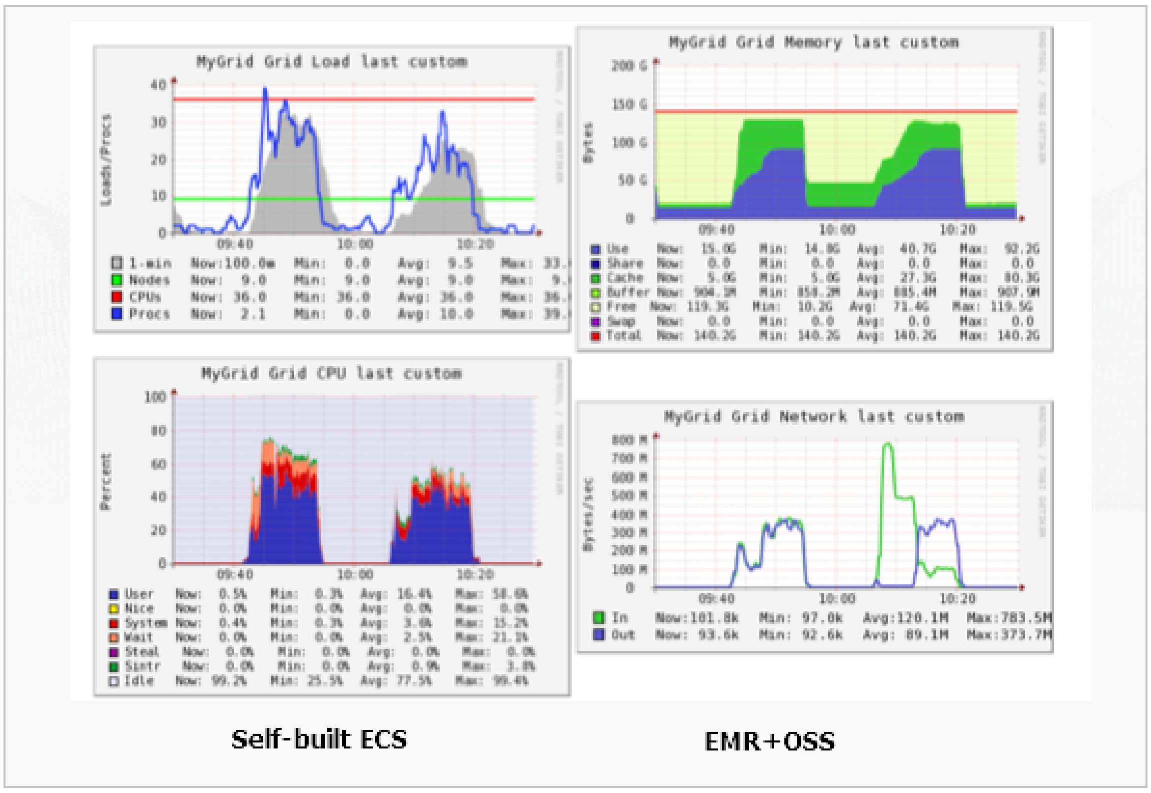
  The Spark test script is as follows.

```
/ opt / apps / spark - 1 . 6 . 1 - bin - hadoop2 . 7 / bin / spark -
 submit  -- master   yarn  -- deploy - mode   cluster  -- executor
 - memory   3G  -- num - executors   30  -- conf   spark . default
 . parallelis  m = 800  -- class   com . github . ehiggs . spark
 . terasort . TeraSort   spark - terasort - 1 . 0 - jar - with -
```

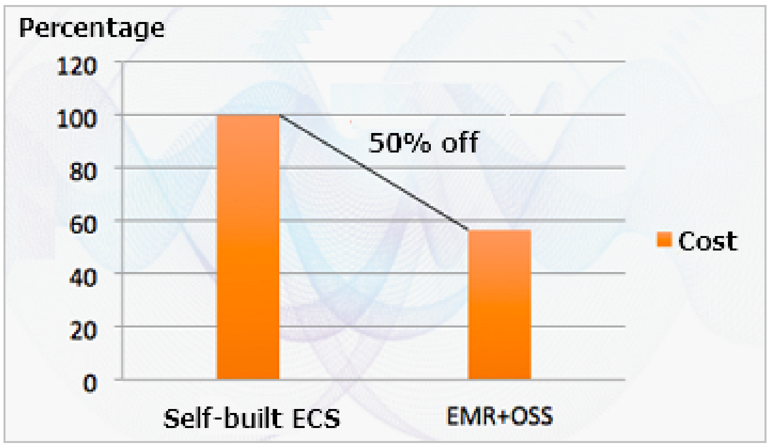
```
dependenci  es . jar  / data / teragen_10  0g  / data / terasort_o
ut_100g
```

- **Test results**

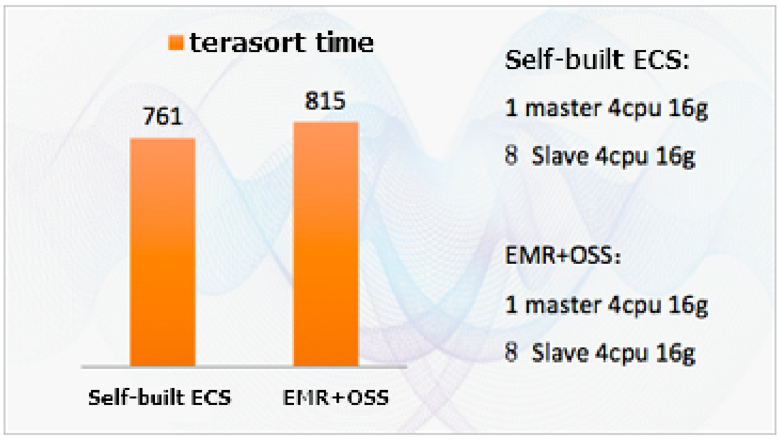
  - Performance



  - Cost



  - Time



- Result analysis

  From the performance chart, we can compare the respective advantages of the EMR + OSS and self-built Hadoop with ECS systems:

  - The overall load is lower
  - Memory utilization is basically the same
  - CPU usage is lower, in which case, the usage level for iowait and sys is much lower. Because the datanode and disk operations of the self-built ECS system occupy resources, this adds to the CPU overhead.
  - In terms of network usage, because sortbenchmark performs two data read operations (the first for sampling and the second for actually reading the data), network usage starts out high, and then in the shuffle+ results output stage, drops to about half of the self-built Hadoop with ECS system. Therefore, from the network perspective, the overall usage is basically flat.

  In short, with EMR + OSS, the cost is halved, but the drop in performance is negligible. Moreover, an increase in the concurrency of the EMR + OSS solution means better time advantage in comparison with the self-built Hadoop with ECS system.

Unsuitable scenarios

We recommend that you do not use EMR + OSS in the following scenarios:

- Scenarios with a large number of small files

  In this case, merge files smaller than 10 MB. The EMR + OSS solution provides the best performance when data volumes exceed 128 MB.
- Scenarios with frequent OSS metadata operations

5 Data backup and recovery

5.1 Back up buckets

Alibaba Cloud offers multiple backup methods for data on OSS to suit different scenarios.

The following methods can be used to back up OSS data on the cloud:

- Cross-region replication (set on the console or using APIs or SDK code)
- ossimport tool

Back up data using cross-region replication

- Applicable scenarios

  See Cross-Region replication development guide.
- Operation on the console

  See Cross-Region replication operation guide.
- FAQ

  See How to synchronize data to OSS.

Note:

- The source bucket and target bucket belong to the same user but different regions.
- The source bucket and target bucket do not use archive storage.
- Data synchronization between buckets in the same region can be implemented using OSS SDK/API code.

Back up data using the ossimport tool

- advantage

  The ossimport tool can migrate data stored on local hosts or other cloud storage systems to OSS. It has the following features:

  - Supports a vast variety of data sources, including local drives, Qiniu Cloud, Baidu BOS, AWS S3, Azure Blob, Blob, but also cloud, Tencent cloud cos, Golden Mountain ks3, HTTP, OSS, and so on, and can be expanded as needed.
  - Supports resumable upload.
  - Supports throttling.
  - Supports migration of objects generated after a specified time or with a specified prefix.
  - Supports parallel data upload and download.
  - Supports the standalone and distributed modes. The standalone mode is easy to deploy and use, while the distributed mode is suitable for large-scale data migration.
- Installation and deployment

  See Architecture and configuration, Standalone deployment , and Distributed deployment.
- FAQ

  See FAQ.
- NOTE

  - If data needs to be migrated between buckets of different user accounts and the data volume exceeds 10 TB, the distributed version is recommended.
  - When using the incremental mode to synchronize object changes between OSS buckets, note that ossimport can synchronize only modification operations (put /append/multipart) and cannot synchronize read or delete operations. No SLA guarantee is provided for timely data synchronization in this mode. Therefore, use the incremental mode with caution.
  - Cross-region replication is recommended for data synchronization between different regions, if cross-region replication is enabled in these regions.

6 Bucket management

6.1 Cross-origin resource sharing (CORS)

Same-origin policy

Cross-origin access, or cross-origin of JavaScript, is a type of browser restriction for security consideration, namely, the same-origin policy. When Website A tries to use the JavaScript code on its webpage to access Website B, the attempt is rejected by the browser because A and B are two websites of different origins.

However, cross-origin access is a commonly used on a day-to-day basis. For example, OSS is used at the backend for the website www.a.com. The JavaScript-based upload function is provided on the webpage. However, requests on the webpage are only sent to www.a.com, whereas all requests sent to other websites are rejected by the browser. As a result, user-uploaded data must be relayed to other sites through www.a.com. If cross-origin access is configured, data can be uploaded directly to OSS instead of relaying it through www.a.com.

CORS overview

CORS is a standard cross-origin solution provided by HTML5. For the specific CORS rules, see W3C CORS Norms.

CORS is a set of control policies followed by the browsers, which use HTTP headers for interaction. When identifying a request initiated as a cross-origin request, a browser adds the Origin header to the HTTP request and sends the request to the server. In the preceding example, the Origin header is www.a.com. After receiving the request, the server determines based on certain rules whether to permit the request. If the request is permitted, the server attaches the Access-Control-Allow-Origin header to the response. The header contains www.a.com, indicating that cross-origin access is allowed. In case, server permits all cross-origin requests, set the Access-Control-Allow-Origin header to *. The browser determines whether the cross-origin request is successful based on whether the corresponding header is returned. If the corresponding header is not attached, the browser blocks the request. This is only a simple description.

The preceding content is a simple scenario. CORS norms classify requests into two types: simple requests and precheck requests. Precheck is a protection mechanism that prevents unauthorized requests from modifying resources. Before sending the actual request, the browser sends an OPTIONS HTTP request to determine whether the server permits the cross-origin request. If the request is not permitted, the browser rejects the actual request.

No precheck request is required only if both of the following conditions are met:

- The request method is one of the following:
  - GET
  - HEAD
  - POST
- All headers are in the following lists:
  - Cache-Control
  - Content-Language
  - Content-Type
  - Expires
  - Last-Modified
  - Pragma

Precheck requests provide information about the subsequent request to the server, that includes:

- Origin: Request origin information.
- Access-Control-Request-Method: Type of the subsequent request, for example, POST or GET.
- Access-Control-Request-Headers: List of headers explicitly set and included in the subsequent request.

After receiving the precheck request, the server determines whether to permit the cross-origin request based on the attached information. The return information is also sent using the following headers:

- Access-Control-Allow-Origin: list of permitted origins for cross-origin requests.
- Access-Control-Allow-Methods: List of permitted cross-origin request methods.
- Access-Control-Allow-Headers: List of permitted cross-origin request headers.

- Access-Control-Expose-Headers: List of headers permitted to be exposed to JavaScript code.
- Access-Control-Max-Age: Maximum browser cache time in seconds.

Based on the returned information, the browser determines whether to send the actual request. If none of these headers is received, the browser rejects the subsequent request.

Note:
The preceding actions are performed automatically by the browser, and you can ignore the details. If the server is correctly configured, the process is the same for non-cross-origin requests.

Scenarios

Access permission control applies to browsers rather than servers, CORS is only applicable in scenarios where a browser is used. Hence, you do not need to worry about cross-origin issues when using other clients.

Applications that use CORS primarily, use Ajax in a browser to directly access OSS, instead of requiring traffic to be redirected through application servers. This applies to the upload and download processes. For websites powered by both OSS and Ajax technology, CORS is recommended for direct communication with OSS.

OSS support for CORS

OSS supports CORS rule configuration for permitting or rejecting corresponding cross-origin requests as required. CORS rules are configured at the bucket level. For more information, see PutBucketCORS.

Whether a CORS request is permitted is independent of OSS identity verification. That is, the OSS CORS rules are only used to determine whether to attach relevant CORS headers. Whether the request is blocked is only determined by the browser.

When using cross-origin requests, pay attention to whether the browser cache is enabled. For example, the same cross-origin resource is requested respectively by two webpages in the same browser (originated from www.a.com and www.b.com) at the same time. If the request of www.a.com is received by the server in the first place, the server returns the resource with the Access-Control-Allow-Origin header "www.a.com". When www.b.com initiates its request, the browser returns its previous cached request. As the header content does not match the CORS request, the subsequent request fails.

Note:
Currently, all OSS object-related interfaces provide CORS verification. In addition, multipart interfaces fully support CORS verification.

Cross-origin GET request example

In this example, Ajax is used to retrieve data from OSS. For simplified description, all used buckets are public. The CORS configuration for accessing a private bucket is the same and only requires a signature to be attached to the request.

Getting started

Create a bucket. For example, create the bucket oss-cors-test with the access right set to public-read. Then create the text file named test.txt, and upload it to the bucket.

Click here to get access address of test.txt .

Note:
Replace the following address with your test address.

Use curl to directly access the file:

```
curl   http :// oss - cors - test . oss - cn - hangzhou . aliyuncs .
com / test . txt
just   for   test
```

The file can be accessed properly.

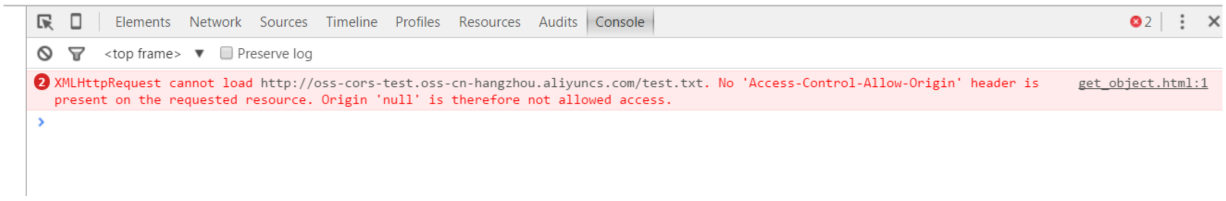
The following code describes how to directly access this website using Ajax. It is the simplest HTML code for access. You can copy the following code, save it as a local HTML file, and open it through your browser. Because no custom headers and hence are included, this request does not require a precheck.

```
<!  DOCTYPE   html >
< html >
< head >
< script   type =" text / javascript "  src ="./ functions . js "></
 script >
</ head >
< body >
< script   type =" text / javascript ">
//  Create   the   XHR   object .
 function   createCORS  Request ( method ,  url ) {
   var   xhr  =  new   XMLHttpReq  uest ();
   if  (" withCreden  tials "  in   xhr ) {
    //  XHR   for   Chrome / Firefox / Opera / Safari .
```

```
      xhr . open ( method ,  url ,  true );
   }  else   if  ( typeof   XDomainReq  uest  ! = " undefined ") {
     //  XDomainReq  uest   for   IE .
      xhr  =  new   XDomainReq  uest ();
      xhr . open ( method ,  url );
   }  else  {
     //  CORS   not   supported .
      xhr  =  null ;
   }
    return   xhr ;
 }
 //  Make   the   actual   CORS   request .
  function   makeCorsRe  quest () {
   //  All   HTML5   Rocks   properties   support   CORS .
    var   url  = ' http :// oss - cors - test . oss - cn - hangzhou .
  aliyuncs . com / test . txt ';
    var   xhr  =  createCORS  Request (' GET ',  url );
    if  (!  xhr ) {
      alert (' CORS   not   supported ');
      return ;
   }
   //  Response   handlers .
    xhr . onload  =  function () {
      var   text  =  xhr . responseTe  xt ;
      var   title  =  text ;
      alert (' Response   from   CORS   request   to  ' +  url  + ': '
  +  title );
   };
    xhr . onerror  =  function () {
      alert (' Woops ,  there   was   an   error   making   the
  request .');
   };
    xhr . send ();
 }
 </ script >
 < p   align =" center "  style =" font - size :  20px ;">
 < a   href ="#"  onclick =" makeCorsRe  quest ();  return   false ;">
  Run   Sample </ a >
 </ p >
 </ body >
 </ html >
```

After opening the file, click the link (Chrome is used in this example). Check that the link cannot be accessed.

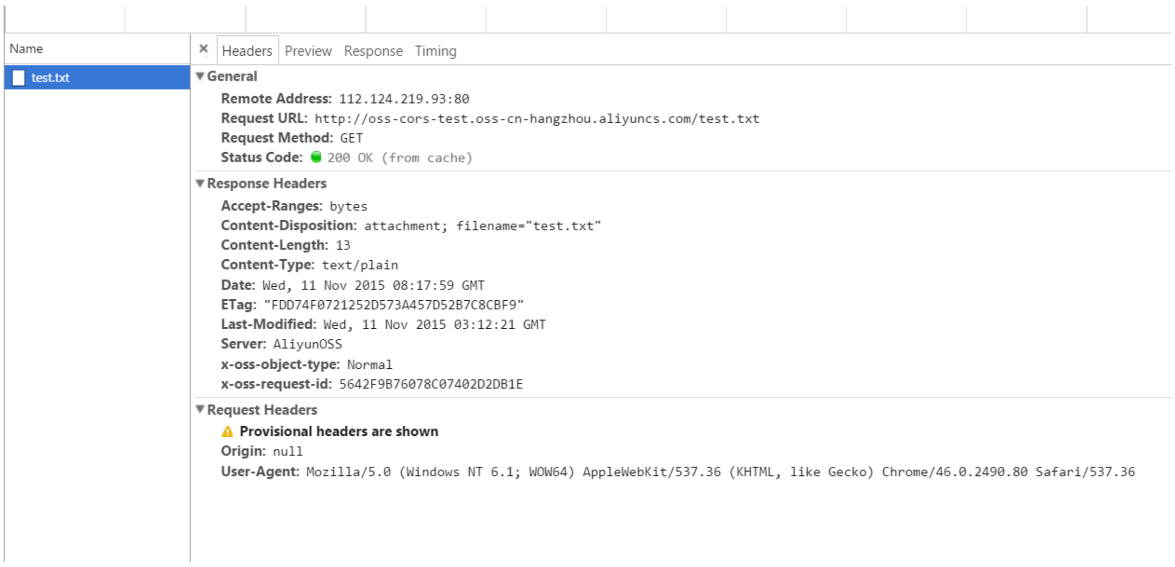
Use Chrome developer tools to identify the cause of the error.



```
XMLHttpRequest cannot load http://oss-cors-test.oss-cn-hangzhou.aliyuncs.com/test.txt. No 'Access-Control-Allow-Origin' header is     get_object.html:1
present on the requested resource. Origin 'null' is therefore not allowed access.
```

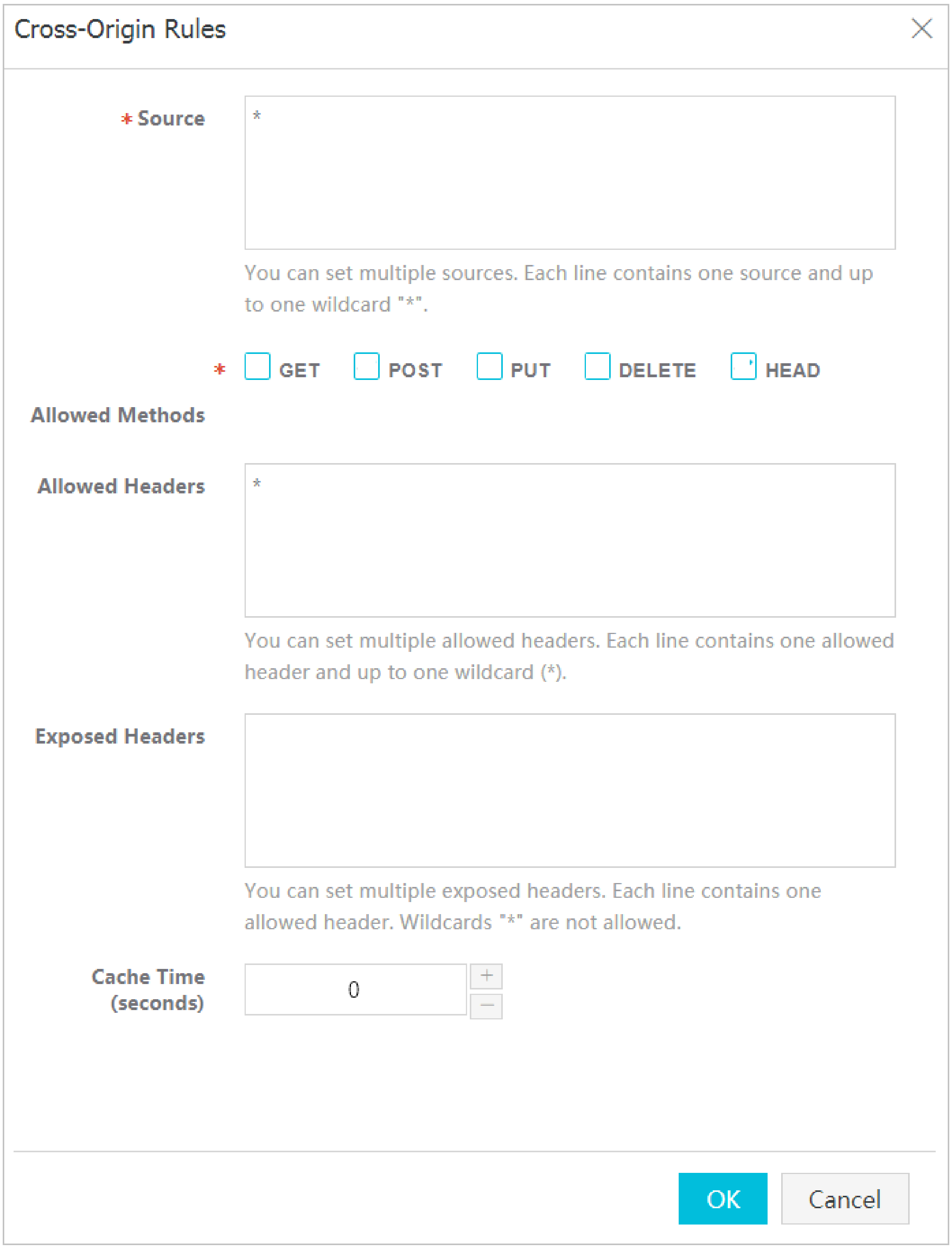
The error is due to the fact that no Access-Control-Allow-Origin header is found. This is because the server is not configured with CORS.

Return to the header interface to check that the browser sends a request with an Origin header. Hence, the request is a cross-origin request. On Chrome, the origin is null because the file is a local file.



```
▼General
    Remote Address: 112.124.219.93:80
    Request URL: http://oss-cors-test.oss-cn-hangzhou.aliyuncs.com/test.txt
    Request Method: GET
    Status Code: ● 200 OK (from cache)
▼Response Headers
    Accept-Ranges: bytes
    Content-Disposition: attachment; filename="test.txt"
    Content-Length: 13
    Content-Type: text/plain
    Date: Wed, 11 Nov 2015 08:17:59 GMT
    ETag: "FDD74F0721252D573A457D52B7C8CBF9"
    Last-Modified: Wed, 11 Nov 2015 03:12:21 GMT
    Server: AliyunOSS
    x-oss-object-type: Normal
    x-oss-request-id: 5642F9B76078C07402D2DB1E
▼Request Headers
    ⚠ Provisional headers are shown
    Origin: null
    User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36
```

Once the problem is located, you can configure CORS settings for the bucket to make sure successful execution of the preceding operation attempt. To facilitate understanding, the following describes how to configure CORS settings on the console. We recommend that CORS be configured on the console if CORS settings are not complex.



Cross-Origin Rules

* Source: *

You can set multiple sources. Each line contains one source and up to one wildcard "*".

* Allowed Methods: GET POST PUT DELETE HEAD

Allowed Headers: *

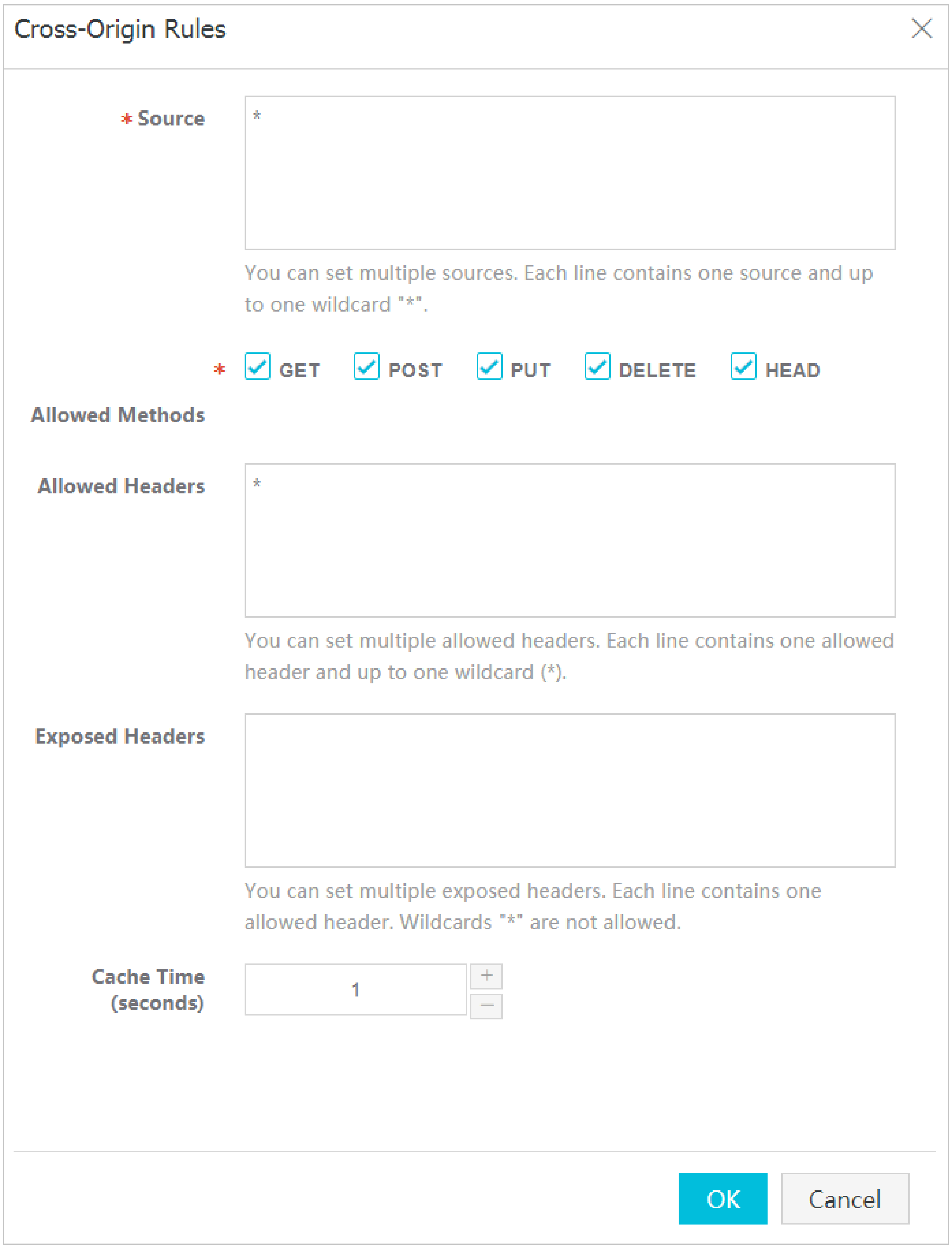
You can set multiple allowed headers. Each line contains one allowed header and up to one wildcard (*).

Exposed Headers:

You can set multiple exposed headers. Each line contains one allowed header. Wildcards "*" are not allowed.

Cache Time (seconds): 0

CORS settings are composed of individual rules. When the system looks for matches , each rule is checked as a match starting with the first rule. The first matched rule applies. The following shows how to add a rule with the loosest configuration:



Cross-Origin Rules

*Source: *

You can set multiple sources. Each line contains one source and up to one wildcard "*".

*Allowed Methods: GET [✓] POST [✓] PUT [✓] DELETE [✓] HEAD [✓]

Allowed Headers: *

You can set multiple allowed headers. Each line contains one allowed header and up to one wildcard (*).

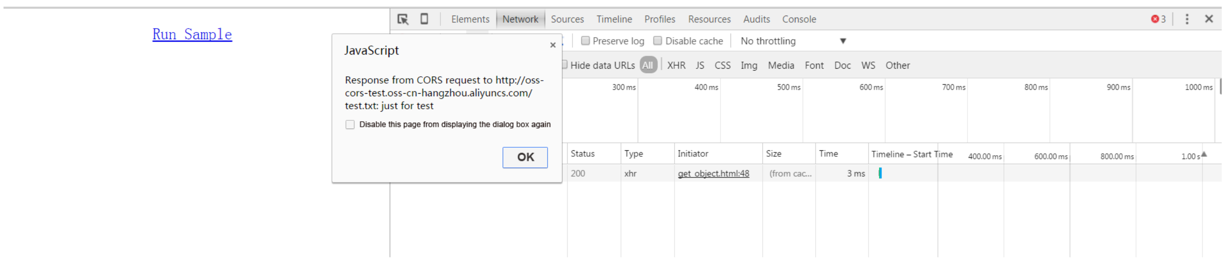
Exposed Headers:

You can set multiple exposed headers. Each line contains one allowed header. Wildcards "*" are not allowed.

Cache Time (seconds): 1

This indicates that access is permitted to all origins, all request types, and all headers , and the maximum cache time is 1s.

Once the configuration is completed, perform the test again. The result is as follows:

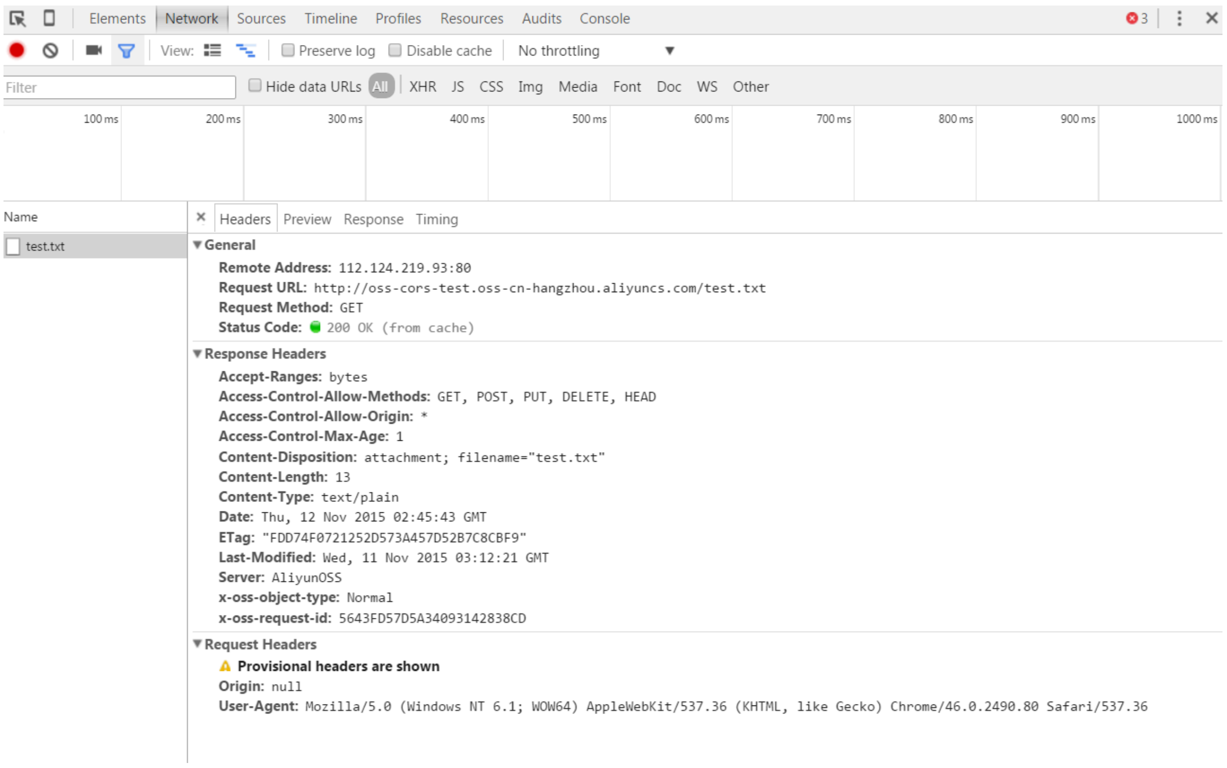

Run Sample

JavaScript

Response from CORS request to http://oss-cors-test.oss-cn-hangzhou.aliyuncs.com/test.txt: just for test

Disable this page from displaying the dialog box again

```
Status  Type  Initiator            Size          Time
200     xhr   get_object.html:48   (from cac...  3 ms
```

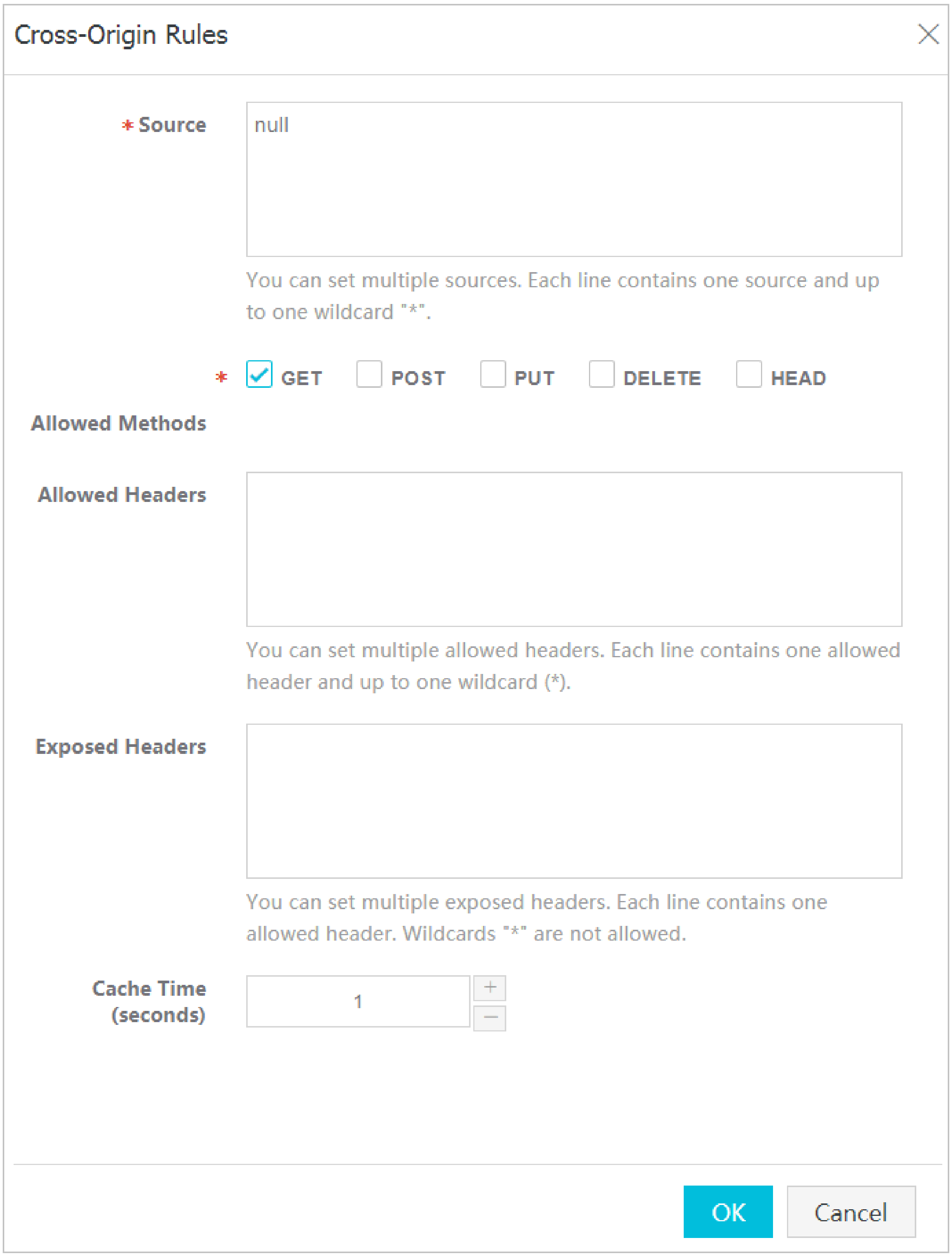

Name: test.txt

```
▼General
    Remote Address: 112.124.219.93:80
    Request URL: http://oss-cors-test.oss-cn-hangzhou.aliyuncs.com/test.txt
    Request Method: GET
    Status Code: 200 OK (from cache)
▼Response Headers
    Accept-Ranges: bytes
    Access-Control-Allow-Methods: GET, POST, PUT, DELETE, HEAD
    Access-Control-Allow-Origin: *
    Access-Control-Max-Age: 1
    Content-Disposition: attachment; filename="test.txt"
    Content-Length: 13
    Content-Type: text/plain
    Date: Thu, 12 Nov 2015 02:45:43 GMT
    ETag: "FDD74F0721252D573A457D52B7C8CBF9"
    Last-Modified: Wed, 11 Nov 2015 03:12:21 GMT
    Server: AliyunOSS
    x-oss-object-type: Normal
    x-oss-request-id: 5643FD57D5A34093142838CD
▼Request Headers
    ⚠ Provisional headers are shown
    Origin: null
    User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36
```

Access requests can be sent properly.

If you are required to troubleshoot cross-origin access problems, you can configure CORS as shown in the preceding figure. This configuration permits all cross-origin requests. If an error occurs under this configuration, the error is not related to CORS.

Besides the loosest configuration, a more refined control mechanism can be configured for targeted control. The following shows the strictest configuration for a successful match:


Cross-Origin Rules

* Source: null

You can set multiple sources. Each line contains one source and up to one wildcard "*".

* Allowed Methods: [✓ GET] POST PUT DELETE HEAD

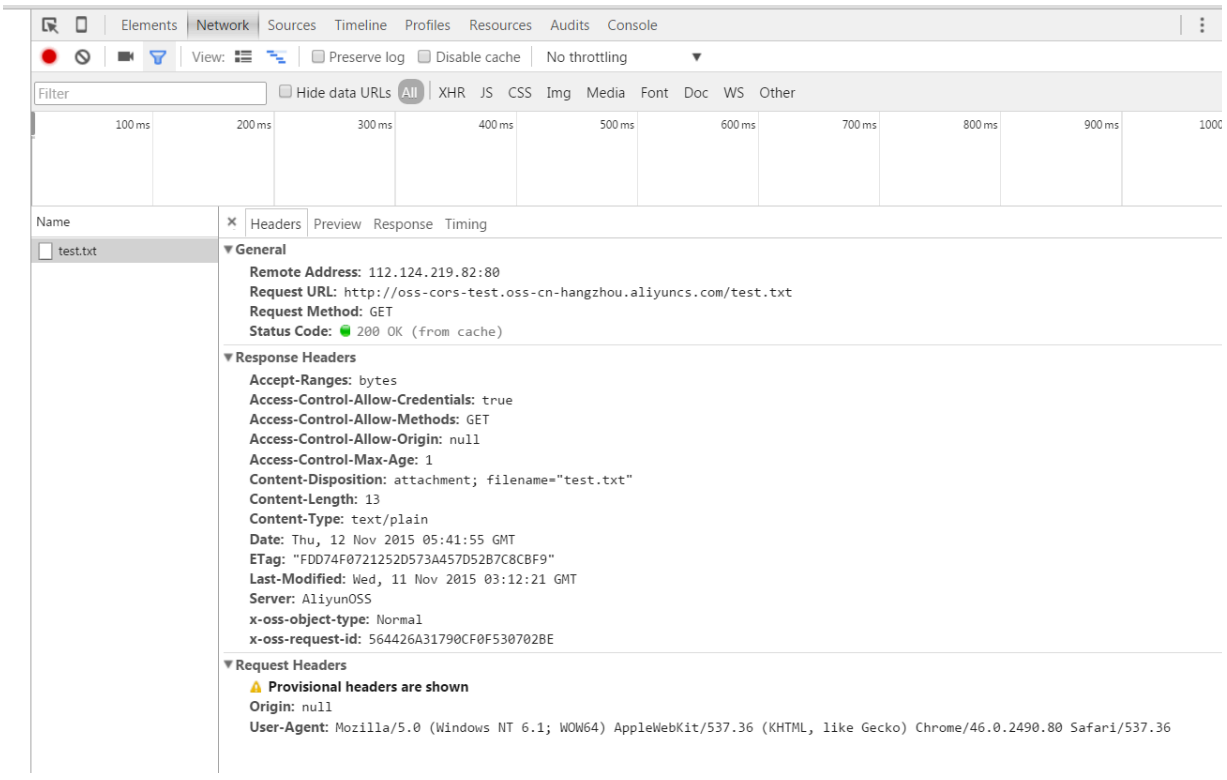
Allowed Headers:

You can set multiple allowed headers. Each line contains one allowed header and up to one wildcard (*).

Exposed Headers:

You can set multiple exposed headers. Each line contains one allowed header. Wildcards "*" are not allowed.

Cache Time (seconds): 1



```
General
    Remote Address: 112.124.219.82:80
    Request URL: http://oss-cors-test.oss-cn-hangzhou.aliyuncs.com/test.txt
    Request Method: GET
    Status Code: 200 OK (from cache)
Response Headers
    Accept-Ranges: bytes
    Access-Control-Allow-Credentials: true
    Access-Control-Allow-Methods: GET
    Access-Control-Allow-Origin: null
    Access-Control-Max-Age: 1
    Content-Disposition: attachment; filename="test.txt"
    Content-Length: 13
    Content-Type: text/plain
    Date: Thu, 12 Nov 2015 05:41:55 GMT
    ETag: "FDD74F0721252D573A457D52B7C8CBF9"
    Last-Modified: Wed, 11 Nov 2015 03:12:21 GMT
    Server: AliyunOSS
    x-oss-object-type: Normal
    x-oss-request-id: 564426A31790CF0F530702BE
Request Headers
    Provisional headers are shown
    Origin: null
    User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36
```

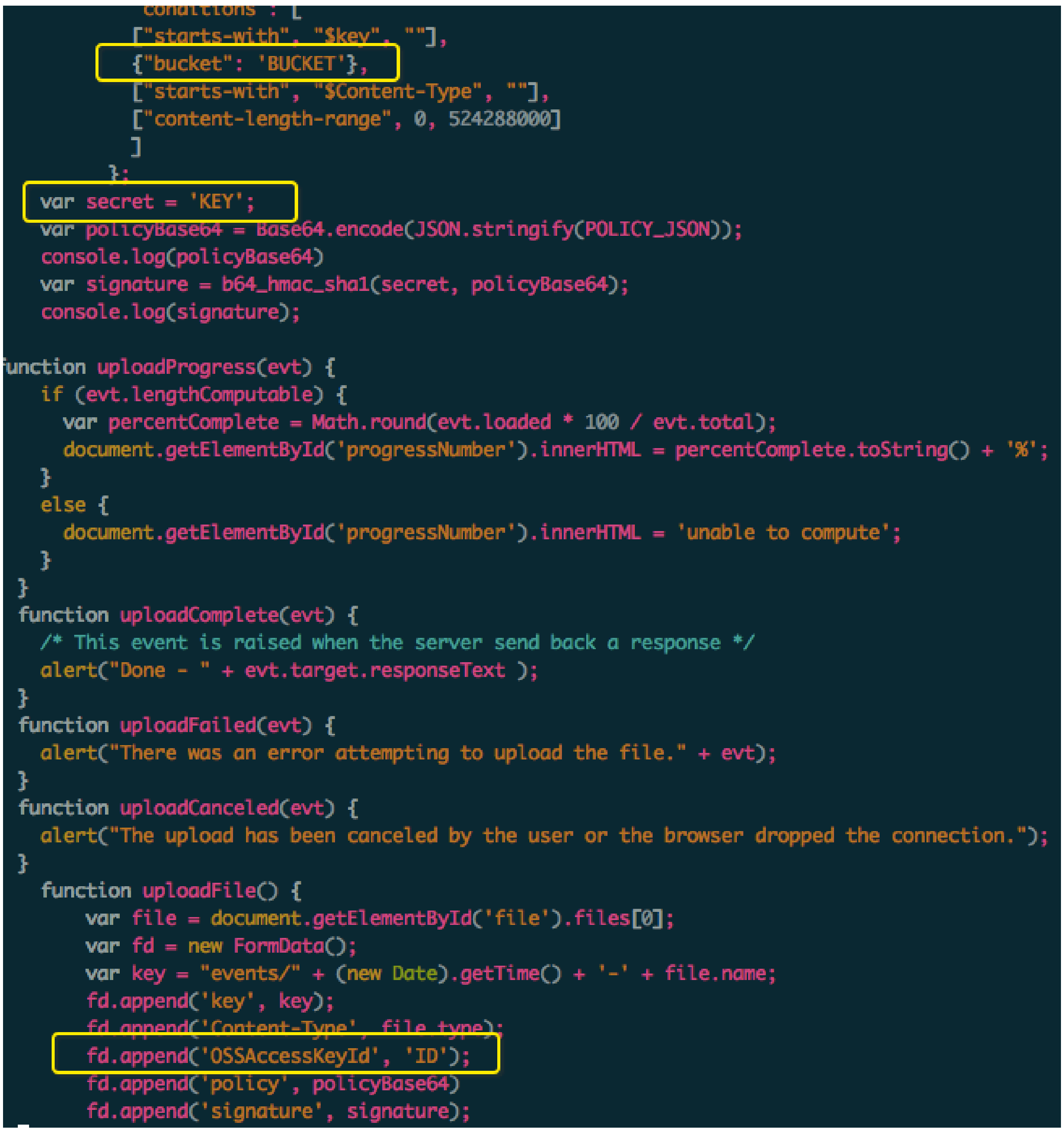
In most cases, we recommend that you use the strictest configuration applicable in their use scenarios to guarantee maximum security at minimal configuration.

Use cross-origin requests for POST upload

The following provides a more complex example where a POST request with a signature is used, and the browser must send a precheck request.

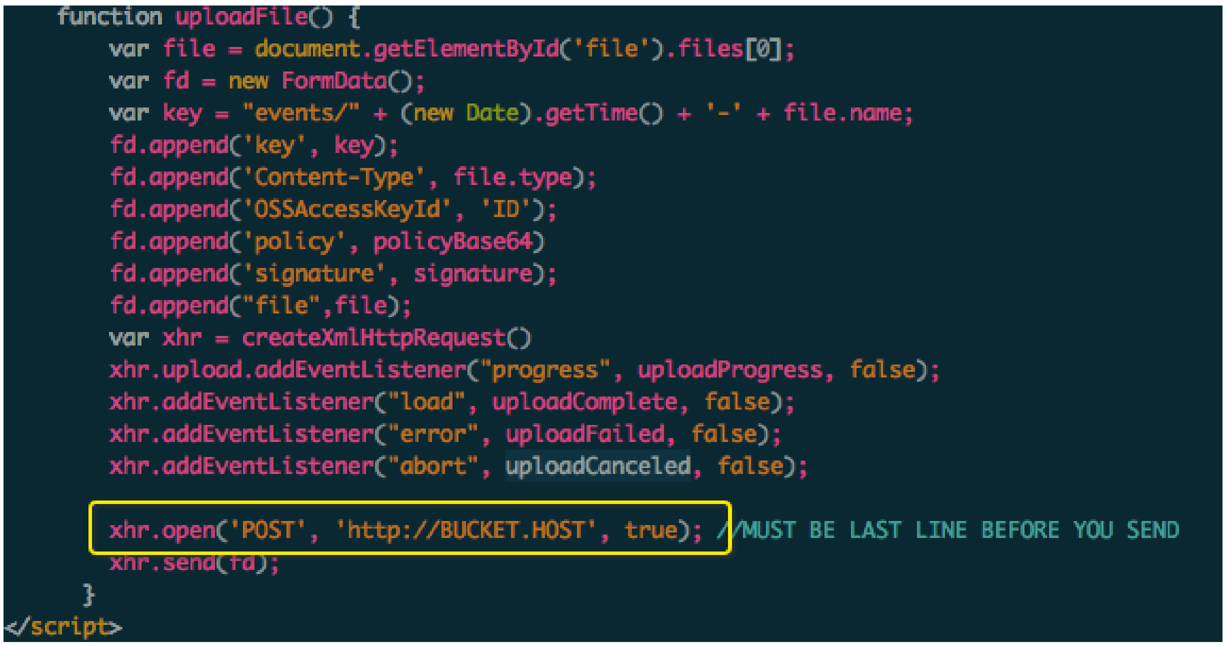
PostObjectSample

Note:
After downloading the preceding code, modify all the following sections to meet your requirements. Then run it on your server.



```
      "conditions": [
      ["starts-with", "$key", ""],
      {"bucket": 'BUCKET'},
      ["starts-with", "$Content-Type", ""],
      ["content-length-range", 0, 524288000]
      ]
     };
  var secret = 'KEY';
  var policyBase64 = Base64.encode(JSON.stringify(POLICY_JSON));
  console.log(policyBase64)
  var signature = b64_hmac_sha1(secret, policyBase64);
  console.log(signature);

unction uploadProgress(evt) {
  if (evt.lengthComputable) {
   var percentComplete = Math.round(evt.loaded * 100 / evt.total);
   document.getElementById('progressNumber').innerHTML = percentComplete.toString() + '%';
  }
  else {
   document.getElementById('progressNumber').innerHTML = 'unable to compute';
  }
 }
 function uploadComplete(evt) {
  /* This event is raised when the server send back a response */
  alert("Done - " + evt.target.responseText );
 }
 function uploadFailed(evt) {
  alert("There was an error attempting to upload the file." + evt);
 }
 function uploadCanceled(evt) {
  alert("The upload has been canceled by the user or the browser dropped the connection.");
 }
  function uploadFile() {
    var file = document.getElementById('file').files[0];
    var fd = new FormData();
    var key = "events/" + (new Date).getTime() + '-' + file.name;
    fd.append('key', key);
    fd.append('Content-Type', file.type);
    fd.append('OSSAccessKeyId', 'ID');
    fd.append('policy', policyBase64)
    fd.append('signature', signature);
```



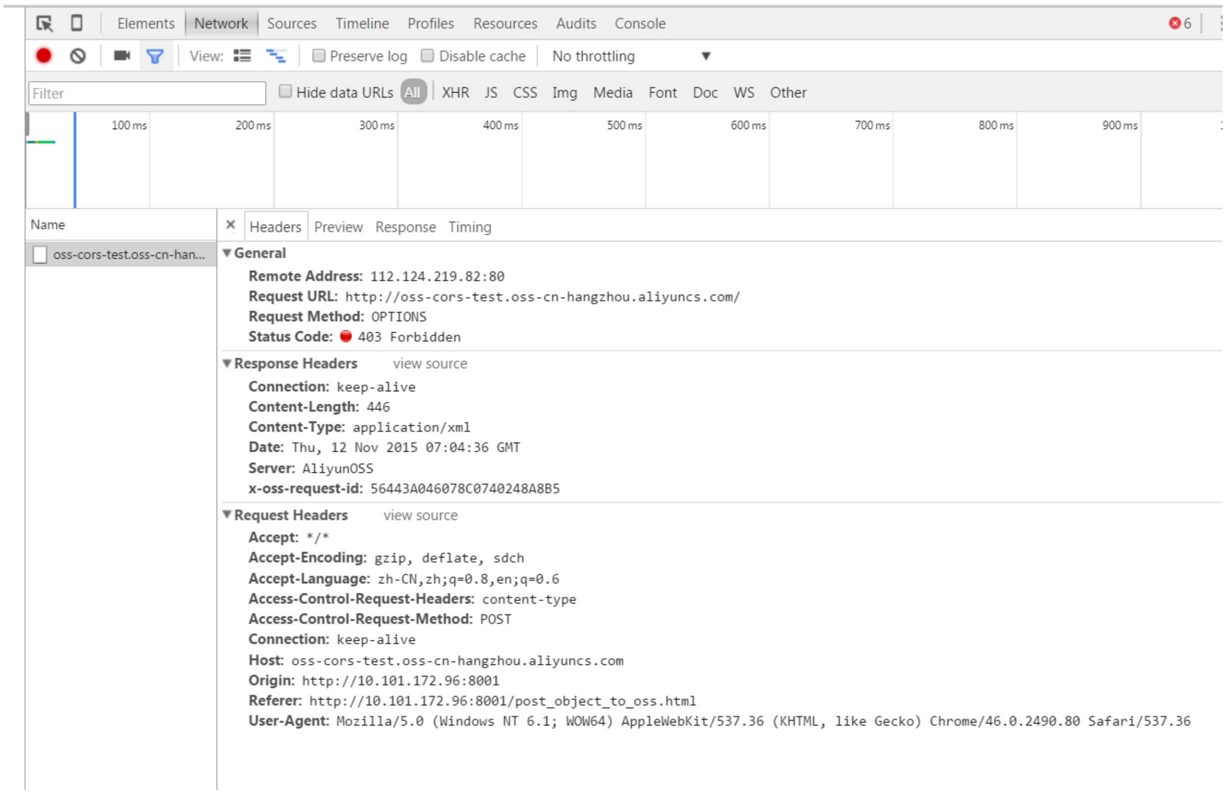
```
    function uploadFile() {
        var file = document.getElementById('file').files[0];
        var fd = new FormData();
        var key = "events/" + (new Date).getTime() + '-' + file.name;
        fd.append('key', key);
        fd.append('Content-Type', file.type);
        fd.append('OSSAccessKeyId', 'ID');
        fd.append('policy', policyBase64)
        fd.append('signature', signature);
        fd.append("file",file);
        var xhr = createXmlHttpRequest()
        xhr.upload.addEventListener("progress", uploadProgress, false);
        xhr.addEventListener("load", uploadComplete, false);
        xhr.addEventListener("error", uploadFailed, false);
        xhr.addEventListener("abort", uploadCanceled, false);

        xhr.open('POST', 'http://BUCKET.HOST', true); //MUST BE LAST LINE BEFORE YOU SEND
        xhr.send(fd);
    }
</script>
```

The following describes how to use the bucket oss-cors-test for testing. Before testing , delete all CORS rules to restore the configuration to its initial state.

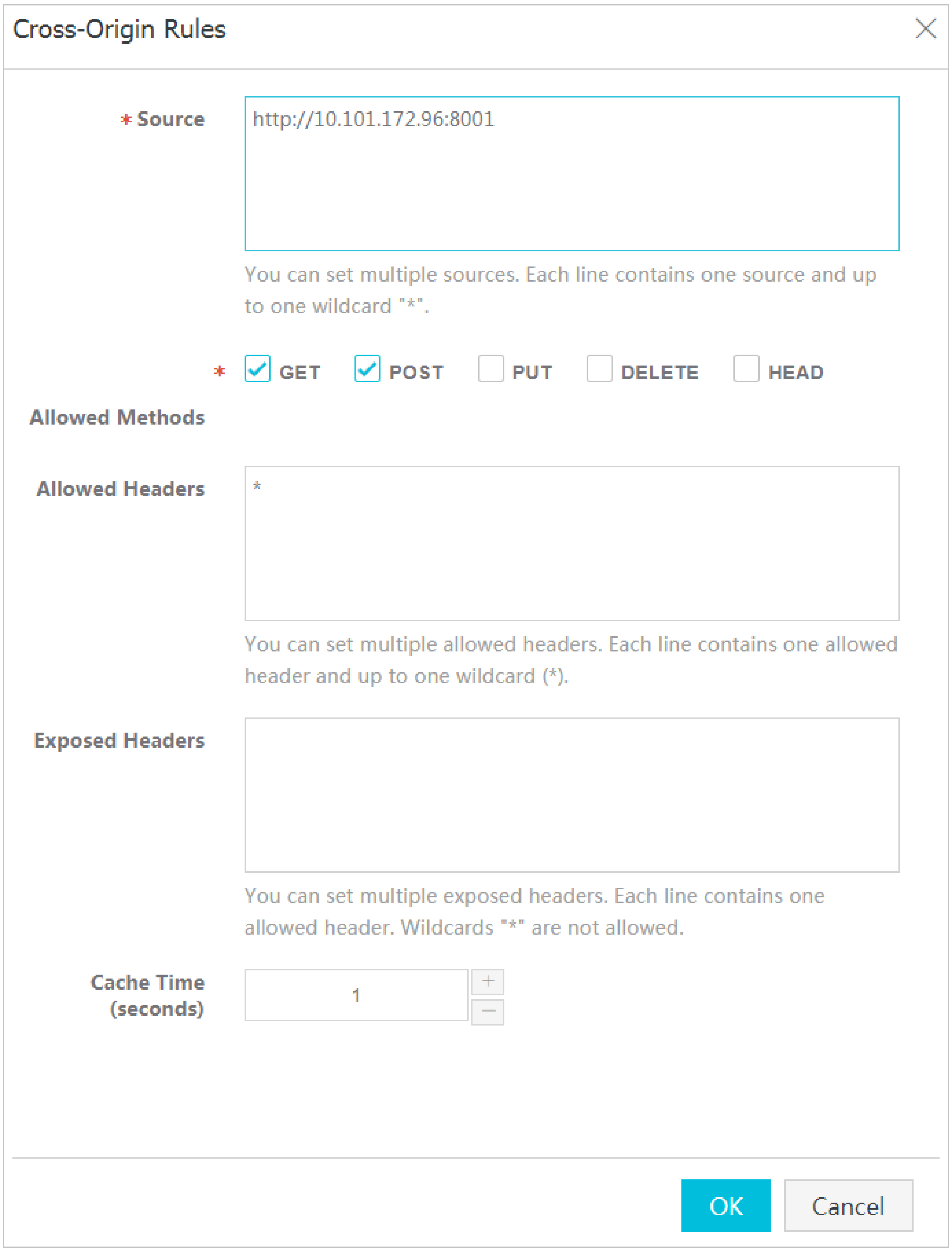
Access this webpage and select a file to upload.

Start the developer tools, and you can view the following content. Based on the previous GET example, it is easy to find the same cross-origin error. Different from the GET request, the request requires a precheck. As shown in the following figure, the operation fails because the OPTIONS response does not have CORS headers.



```
▼General
    Remote Address: 112.124.219.82:80
    Request URL: http://oss-cors-test.oss-cn-hangzhou.aliyuncs.com/
    Request Method: OPTIONS
    Status Code: 403 Forbidden
▼Response Headers      view source
    Connection: keep-alive
    Content-Length: 446
    Content-Type: application/xml
    Date: Thu, 12 Nov 2015 07:04:36 GMT
    Server: AliyunOSS
    x-oss-request-id: 56443A046078C0740248A8B5
▼Request Headers      view source
    Accept: */*
    Accept-Encoding: gzip, deflate, sdch
    Accept-Language: zh-CN,zh;q=0.8,en;q=0.6
    Access-Control-Request-Headers: content-type
    Access-Control-Request-Method: POST
    Connection: keep-alive
    Host: oss-cors-test.oss-cn-hangzhou.aliyuncs.com
    Origin: http://10.101.172.96:8001
    Referer: http://10.101.172.96:8001/post_object_to_oss.html
    User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36
```

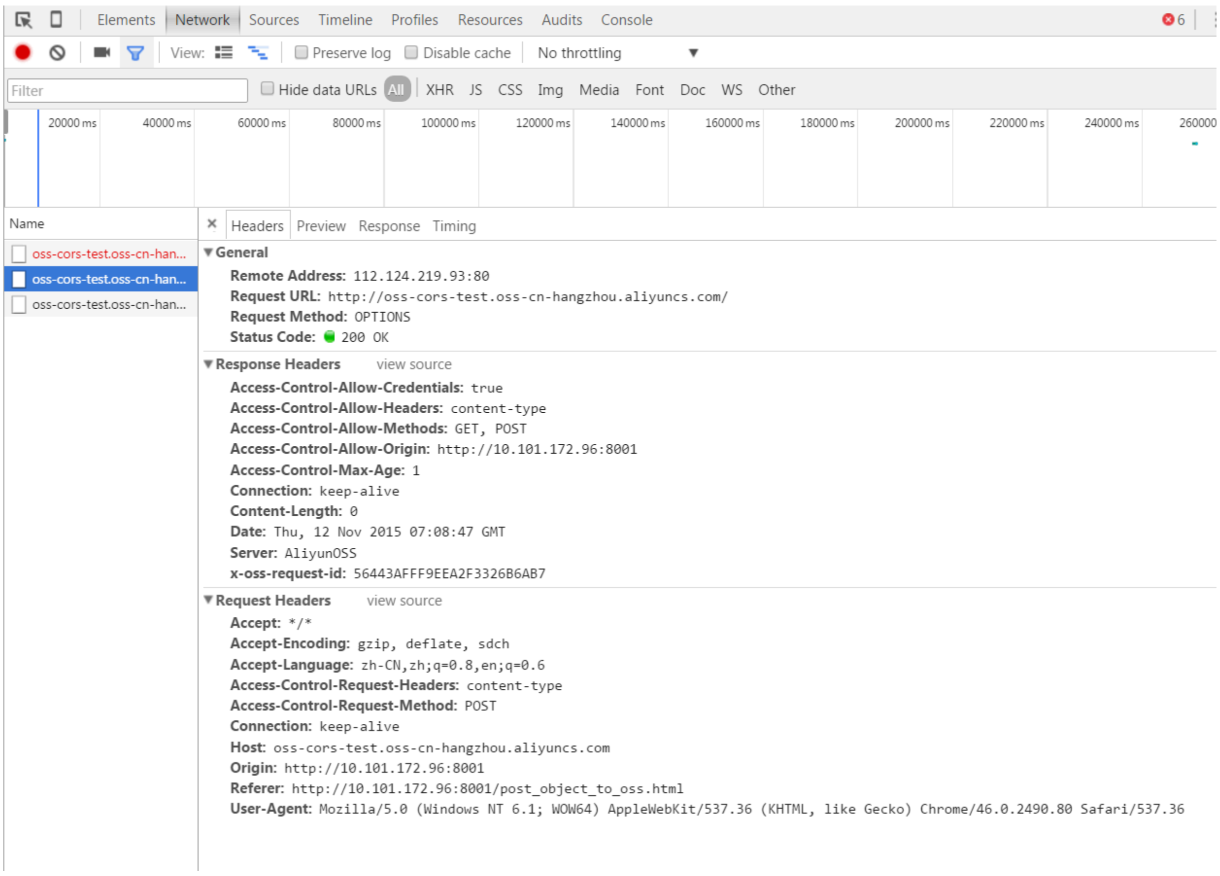
Modify the CORS configuration accordingly.



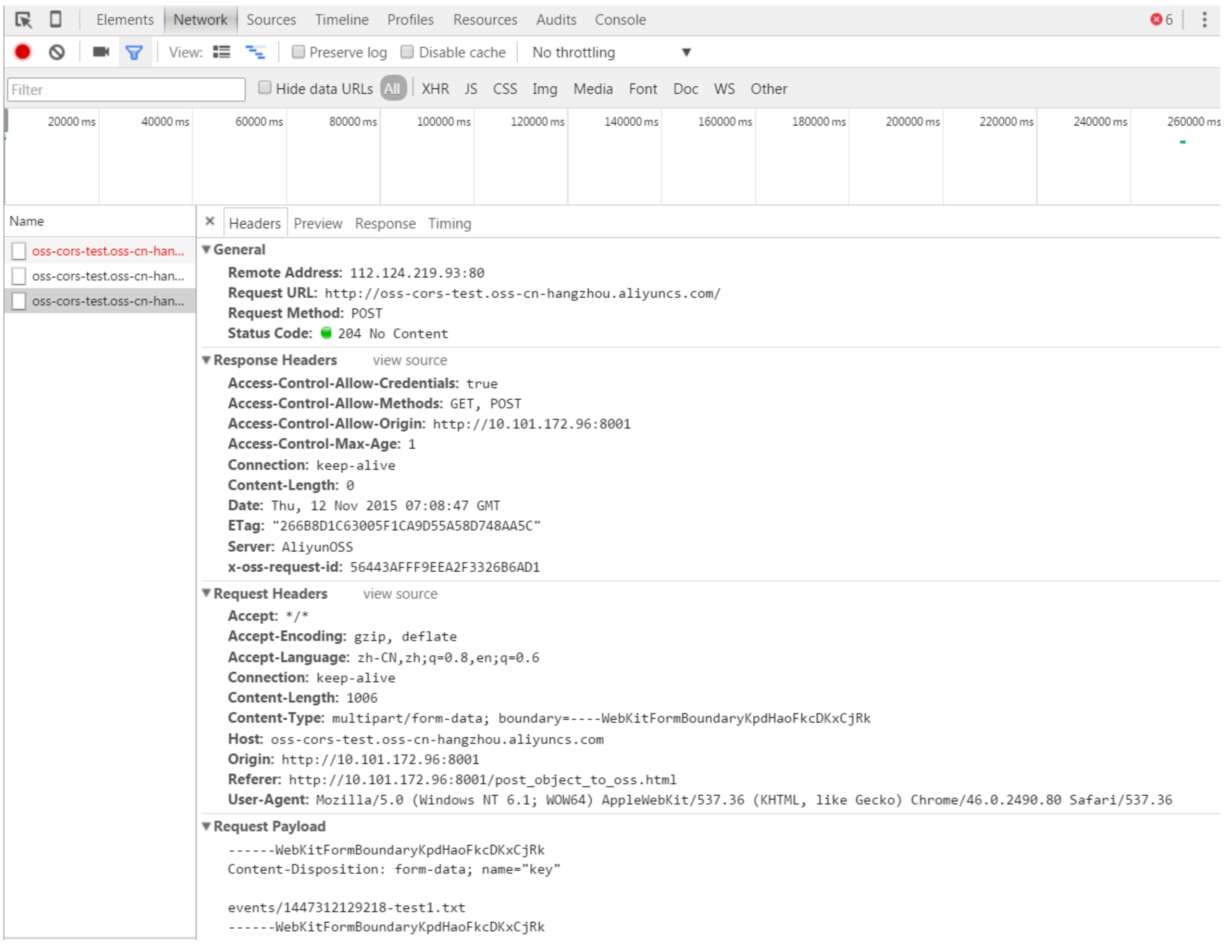
Cross-Origin Rules

* Source	http://10.101.172.96:8001
	You can set multiple sources. Each line contains one source and up to one wildcard "*".
* Allowed Methods	[✓ GET] [✓ POST] ☐ PUT ☐ DELETE ☐ HEAD
Allowed Headers	*
	You can set multiple allowed headers. Each line contains one allowed header and up to one wildcard (*).
Exposed Headers	
	You can set multiple exposed headers. Each line contains one allowed header. Wildcards "*" are not allowed.
Cache Time (seconds)	1

You can perform the operation again to get a successful result. The console displays the newly uploaded file.

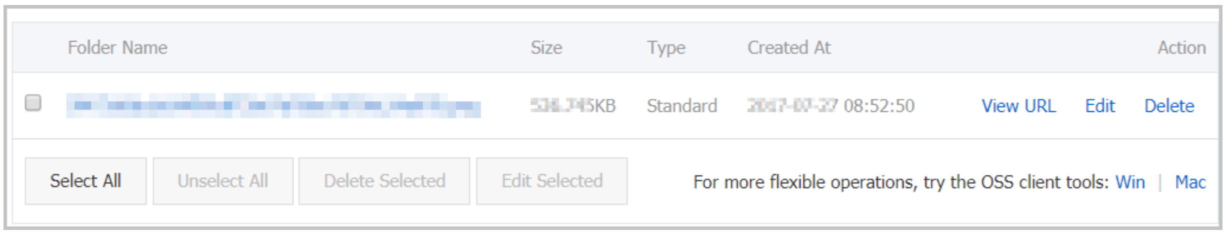


```
▼General
    Remote Address: 112.124.219.93:80
    Request URL: http://oss-cors-test.oss-cn-hangzhou.aliyuncs.com/
    Request Method: OPTIONS
    Status Code: ● 200 OK
▼Response Headers      view source
    Access-Control-Allow-Credentials: true
    Access-Control-Allow-Headers: content-type
    Access-Control-Allow-Methods: GET, POST
    Access-Control-Allow-Origin: http://10.101.172.96:8001
    Access-Control-Max-Age: 1
    Connection: keep-alive
    Content-Length: 0
    Date: Thu, 12 Nov 2015 07:08:47 GMT
    Server: AliyunOSS
    x-oss-request-id: 56443AFFF9EEA2F3326B6AB7
▼Request Headers      view source
    Accept: */*
    Accept-Encoding: gzip, deflate, sdch
    Accept-Language: zh-CN,zh;q=0.8,en;q=0.6
    Access-Control-Request-Headers: content-type
    Access-Control-Request-Method: POST
    Connection: keep-alive
    Host: oss-cors-test.oss-cn-hangzhou.aliyuncs.com
    Origin: http://10.101.172.96:8001
    Referer: http://10.101.172.96:8001/post_object_to_oss.html
    User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36
```



```
▼General
    Remote Address: 112.124.219.93:80
    Request URL: http://oss-cors-test.oss-cn-hangzhou.aliyuncs.com/
    Request Method: POST
    Status Code: ● 204 No Content
▼Response Headers      view source
    Access-Control-Allow-Credentials: true
    Access-Control-Allow-Methods: GET, POST
    Access-Control-Allow-Origin: http://10.101.172.96:8001
    Access-Control-Max-Age: 1
    Connection: keep-alive
    Content-Length: 0
    Date: Thu, 12 Nov 2015 07:08:47 GMT
    ETag: "266B8D1C63005F1CA9D55A58D748AA5C"
    Server: AliyunOSS
    x-oss-request-id: 56443AFFF9EEA2F3326B6AD1
▼Request Headers      view source
    Accept: */*
    Accept-Encoding: gzip, deflate
    Accept-Language: zh-CN,zh;q=0.8,en;q=0.6
    Connection: keep-alive
    Content-Length: 1006
    Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryKpdHaoFkcDKxCjRk
    Host: oss-cors-test.oss-cn-hangzhou.aliyuncs.com
    Origin: http://10.101.172.96:8001
    Referer: http://10.101.172.96:8001/post_object_to_oss.html
    User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36
▼Request Payload
    ------WebKitFormBoundaryKpdHaoFkcDKxCjRk
    Content-Disposition: form-data; name="key"

    events/1447312129218-test1.txt
    ------WebKitFormBoundaryKpdHaoFkcDKxCjRk
```



	Folder Name	Size	Type	Created At	Action
☐	[redacted]	[redacted]KB	Standard	[redacted] 08:52:50	View URL Edit Delete

Select All
Unselect All
Delete Selected
Edit Selected
For more flexible operations, try the OSS client tools: Win | Mac

Test content:

```
$ curl   http :// oss - cors - test . oss - cn - hangzhou . aliyuncs .
 com / events / 1447312129  218 - test1 . txt
 post   object   test
```

CORS configuration caveats

CORS configuration items include:

- Source: Provide the complete domain information during configuration, for example, `http :// 10 . 101 . 172 . 96 : 8001` as shown in the preceding figure.

  Do not omit the protocol name, for example, http. Include the port number if the default one has been changed. If you are not sure, use the browser's debugging function to view the Origin header. This field supports the wildcard *, but only one such symbol can be used. You can perform configuration based on your needs.
- Method: Select the allowed methods based on your requirements.
- Allow Header: Indicates the list of allowed headers. To avoid header omission, we recommend that you set this field to * unless otherwise specified. The header is not case-sensitive.
- Expose Header: Indicates the list of headers exposed to the browser. Wildcards cannot be used. The specific configuration must be selected according to your application. Expose only required headers, for example, ETag headers. If you do not need to expose this information, you can leave this field blank. You can specify headers individually. This field is not case-sensitive.
- Cache Time: In normal cases, you can set a relatively large value, for example, 60s.

The CORS configuration method sets individual rules for each origin that may access the service. If possible, do not include multiple origins in a single rule, and avoid overlap or conflict among multiple rules. For other permissions, you only need to grant the required permissions.

Troubleshooting advice

It is easy to mix up other errors with CORS errors when similar programs are debugged.

For example, when an access request is rejected because of any incorrect signature , the return result may not contain CORS header information because permission verification precedes CORS verification. In this case, some browsers directly report a CORS error, but the actual CORS configuration on the server is correct. The following two methods can be used to address the preceding problem:

- View the HTTP request's return value. Because CORS verification is an independen t process that does not affect core processes, a return value such as 403 is not produced by CORS. You must first rule out the program-related causes. If a precheck request is sent previously, you can view the precheck request results. If the correct CORS headers are returned, the actual request is permitted by the server. Therefore, the error can only be caused by another component.
- Set the server's CORS configuration to the loosest setup shown in the preceding example. Use wildcards to permit all origins and request types. Then re-verify the configuration. If the verification still fails, it is possible that other type of errors have occured.

6.2 Anti-leech

Background

For example, A is the webmaster of a website. Webpages on the website contain links to images and audio/video files. These static resources are stored on Alibaba Cloud OSS. For example, A may save an image file on OSS with the URL `http :// referer - test . oss - cn - hangzhou . aliyuncs . com / aliyun - logo . png`.

For OSS external resource url, see OSS address such a URL (without signing) requires the user's bucket permission to read publicly.

B is the webmaster of another website, B use the image resources of the website without permission, use this method to steal space and traffic by placing it in a web page on your website. In this case, the third-party web site user sees the B web site, but it's not clear the source of the pictures on the website. Since OSS charges by usage , so that user A does not get any benefit, instead, the cost of resource use is borne.

This article applies to users who use OSS resources as outer chains in a Web page, it also introduces a-like users who have stored their resources on OSS, how to avoid the use of unnecessary resources by setting up anti-theft chains.

Implementation method

At present, the methods of anti-theft chain provided by OSS mainly include the following two types:

- Set Referer : The operation is available through the console and the SDK, and the user can choose according to their needs.
- Use signature URL: This is suitable for users who are used to developing.

The following two examples are provided in this article:

- Set the Referer anti-theft chain through the console
- Dynamic generation of signed URL anti-theft chains based on PHP SDK

Set Referer

This section focuses on what Referer is and how OSS uses Referer for anti-theft chains .

- What is Referer?

  Referer is HTTP Part of the header that usually comes with a referer when the browser sends a request to the web server, tell the server the source of the link for this request. In the example above, if the web site for user B is userd omain-steal, want to steal a picture link `http :// referer - test . oss - cn - hangzhou . aliyuncs . com / aliyun - logo . png` . A's website domain name is `userdomain` .

  Suppose the web page of the chain website user domain-steal is as follows:

```
< html >
    < p > This   is   a   test </ p >
    < img   src =" http :// referer - test . oss - cn - hangzhou .
 aliyuncs . com / aliyun - logo . png " />
</ html >
```

  Assume the web page with the source station user domain is as follows:

```
< html >
    < p > This   is   my   test   link   from   OSS   URL </ p >
    < img   src =" http :// referer - test . oss - cn - hangzhou .
 aliyuncs . com / aliyun - logo . png " />
```

```
</ html >
```

- When an Internet user uses a browser to access the Web page of B's website `http :// userdomain - steal / index . html` , the link in the web page is a picture of the site A. Because a request from one domain name (user domain-steal) jumped to another domain name (maid), the browser takes the Referer with it in the header of the HTTP request, as shown:

  You can see that the browser Referer in the HTTP request is `http :// userdomain - steal / index . html` . This article mainly uses Chrome's developer mode to view web page requests, as follows:
- The same browser visits `http :// userdomain / error . html` , and you can also see that the browser's Referer is `http :// userdomain / error . html` .
- If the browser enters the address directly, you can see that Referer is empty in the request.

  If a does not have any Referer-related settings on the OSS, all three cases have access to the picture link for user.

- The principle of OSS through Referer anti-theft chain

  Thus, when the browser requests the OSS resource, if a page Jump occurs, the browser takes the Referer in the request, and the Referer's value is the URL on the previous page, sometimes Referer is empty.

  For both cases, the OSS Referer feature offers two options:

  - Sets whether empty Referer access is allowed. It cannot be set separately and needs to be used in conjunction with the Referer whitelist.
  - Sets the Referer whitelist.

  The details are analyzed as follows:

  - Anti-theft chain authentication is performed only if the user is accessing the object through a signed URL or an anonymous access. If the requested header has an "Authorization" field, it does not do anti-theft chain validation.
  - A bucket can support multiple Referer parameters.
  - The Referer parameter supports wildcard characters '*' and '?'.
  - Users can set up to allow request access for empty referer.
  - When the whitelist is empty, the Referer field is not checked for empty (otherwise all requests will be rejected, because empty Referer will be rejected, for non-empty Referer OSS is also not found on the Referer whitelist).
  - The whitelist is not empty, and a rule is set that does not allow Referer fields to be empty. Only Referer's whitelist of requests is allowed, other requests, including those whose Referer is empty, are rejected.
  - The whitelist is not empty, but the rule "allow Referer field to be empty" is set. An empty request with Referer and a whitelist-compliant request are permitted, other requests are rejected.
  - Three permissions of bucket (private, public-read, public-read-write) the Referer field is checked.

  Wildcard character explanation:

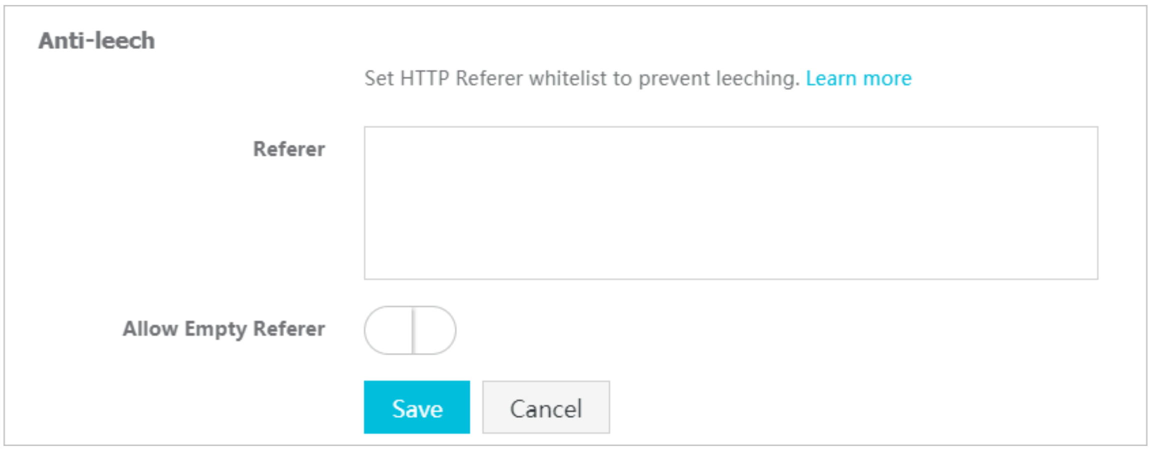
  - Asterisks '*': You can use an asterisks instead of 0 or more characters. If you are looking for a file name that starts with “AEW”, you can enter `AEW` to search for all types of files with the names starting with “AEW”, for example, AEWT.txt, AEWU.EXE, and AEWI.dll. If you want to narrow down the search scope, you can enter `AEW.txt` to search for all .txt files with names starting with AEW, such as AEWIP.txt and AEWDF.txt.

- Question mark (?): represents one character. If you enter love?, all types of files with names starting with "love" and ending with a character are displayed, such as lovey and lovei. If you want to narrow the search scope, you can enter love?.doc to search for all .doc files with names starting with "love" and ending with a character, such as lovey.doc and lovei.doc.

· Anti-leech effects of different Referer settings

The following describes the effects of Referer settings:

- Disable Allow Empty Referer, as shown in the following figure:


Anti-leech

Set HTTP Referer whitelist to prevent leeching. Learn more

Referer

Allow Empty Referer

Direct access: The resources are accessible even when anti-leech protection takes effect. The reason is, if the whitelist is blank, the system does not check whether the Referer field is blank. The Referer setting does not take effect when the whitelist is blank. Therefore, the Referer whitelist must be configured.

- Disable Allow Empty Referer and configure a Referer whitelist.

As shown in the preceding example, the Referer in the browser request is the URL of the current webpage. Therefore, it is necessary to know from which URL the request jumps and then specify the URL.

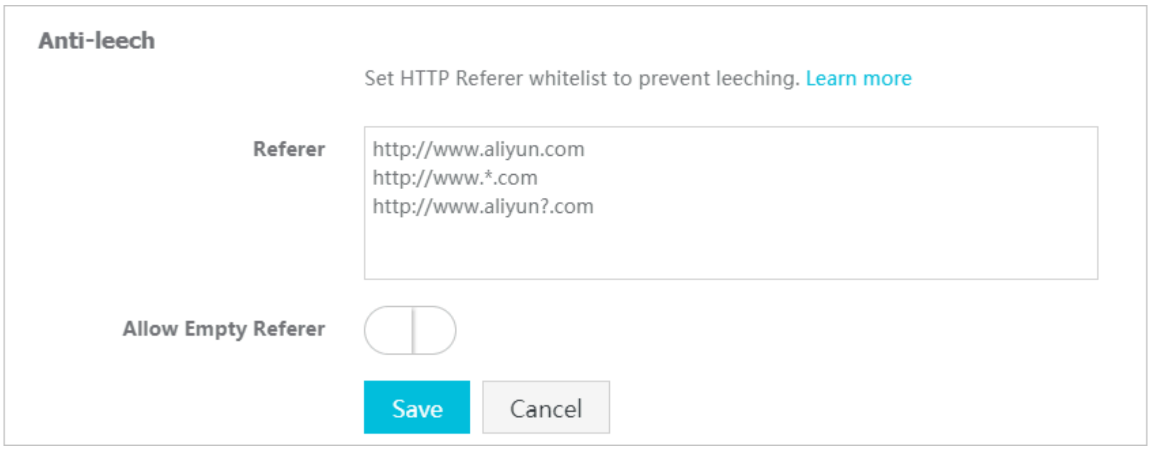
Referer whitelist setting rules:

■ In the example, the Referer is `http :// userdomain / error . html` . Therefore, the Referer whitelist can be set to `http :// userdomain / error . html` . As the Referer check performed by OSS is based on prefix matching, access to other webpages such as `http :// userdomain / index`

`. html` fails. To avoid this problem, you can set the Referer whitelist set to `http :// userdomain /`.

- To allow access to other domain names such as `http :// img . userdomain / index . html`, add `http ://*. userdomain /` to the Referer whitelist.

Both entries are configured as shown in the following figure:


Anti-leech

Set HTTP Referer whitelist to prevent leeching. Learn more

Referer

http://www.aliyun.com
http://www.*.com
http://www.aliyun?.com

Allow Empty Referer

After testing, the following results are obtained:

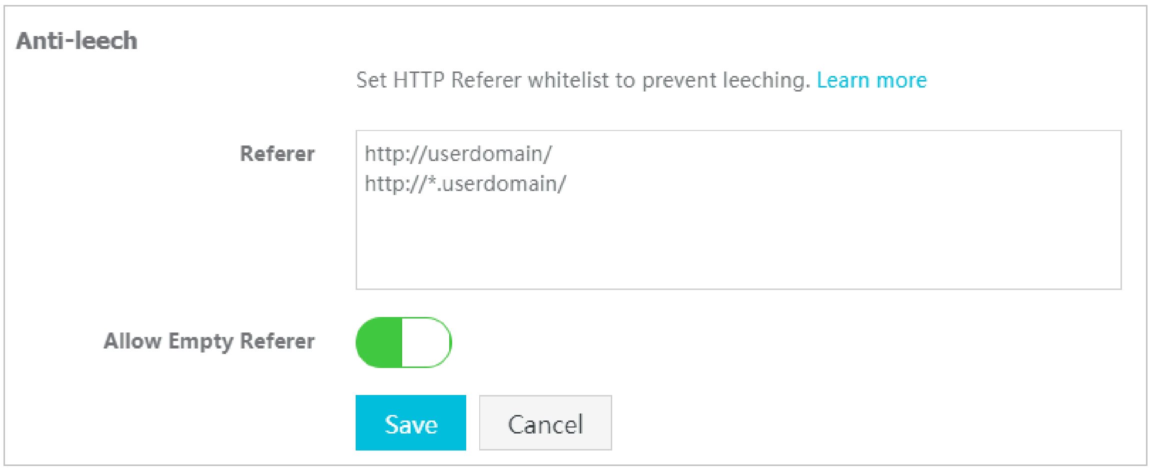
Browser input	Expectation	Result
http://referer-test.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png	Expectation for direct access with a blank Referer: Blank Referers are not allowed and OSS returns 403.	As expected
http://userdomain/error.html	Expectation for a request from the origin site: successful access.	As expected
http://userdomain-steal/index.html	Expectation for a request from a leeching site: OSS returns 403. Anti-leech protection is successful.	As expected
http://img.userdomain/error.html	Expectation for a request from a third-level domain of the origin site : successful access.	As expected

Note:

- In this test, the domain names only serve as examples, and are not the same as the actual domain names you use. Be sure to differentiate them.
- If the Referer whitelist only contains `http :// userdomain /`, and the browser attempts to access the resources through the simulated third-level domain name `http :// img . userdomain / error . html`, the third-level domain name fails to match any of the entries in the Referer whitelist, and OSS returns 403.

- Enable Allow Empty Referer and configure a Referer whitelist.

  The Referer whitelist contains `http ://*. userdomain /` and `http :// userdomain`, as shown in the following figure:



Anti-leech

Set HTTP Referer whitelist to prevent leeching. Learn more

Referer: http://userdomain/
http://*.userdomain/

Allow Empty Referer: on

After testing, the following results are obtained:

Browser input	Expectation	Result
http://referer-test.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png	Expectation for direct access with a blank Referer: successful access	As expected
http://userdomain/error.html	Expectation for a request from the origin site: successful access	As expected
http://userdomain-steal/index.html	Expectation for a request from a leeching site: OSS returns 403. Anti-leech protection is successful.	As expected

Browser input	Expectation	Result
http://img.userdomain/error.html	Expectation for a request from a third-level domain of the origin site : successful access	As expected

- How to configure Referer on OSS

  Functional use reference:

  - API: Put Bucket Referer
  - Console: Anti-leech settings
- Pros and cons of Referer anti-leech protection

  Referer anti-leech protection can be easily configured on the console. The main drawback of the Referer anti-leech protection is that it cannot prevent access attempts by the malicious spoofing Referers. If a leecher uses an application to simulate HTTP requests with a spoofing Referer, the Referer can bypass anti-leech protection settings. If you have higher anti-leech protection requirements, consider using signed URL anti-leech protection.

Signed URLs

For the principles and implementation methods for signed URLs, see Authorizing third-Party download. A signed URL is implemented as follows:

1. Set the bucket permission to private.
2. Generate a signature based on the expected expiration time (the time when the signed URL expires).

Specific implementation

1. Install the latest PHP code by referring to the PHP SDK documentation.
2. Generate a signed URL and add it to the webpage as an external link, for example:

```
<?  php
  require  ' vendor / autoload . php ';
 # Indicates   the   automatic   loading   function   provided   by
   the   latest   PHP .
  use   OSS \ OssClient ;
 # Indicates   the   namespace   used .
 $ accessKeyI  d =" a5etodit71  tlznjt3pdx ****";
 # Indicates   the   AccessKeyI  d ,  which   must   be   replaced
 by   the   one   you   use .
 $ accessKeyS  ecret =" secret_key ";
 # Indicates   the   AccessKeyS  ecret ,  which   must   be
 replaced   by   the   one   you   use .
 $ endpoint =" oss - cn - hangzhou . aliyuncs . com ";
```

```
# Indicates   the   Endpoint ,  selected   based   on   the
region   created   by   the   bucket .  In   the   example ,  the
endpoint   is   Hangzhou .
$ bucket  = ' referer - test ';
# Indicates   the   bucket ,  which   must   be   replaced   by
the   one   you   use .
$ ossClient  =  new   OssClient ($ accessKeyI  d , $ accessKeyS
ecret , $ endpoint );
$ object  = " aliyun - logo . png ";
# Indicates   the   object   to   be   signed .
$ timeout  =  300 ;
# Indicates   the   expected   link   expiration   time .  The
value   indicates   that   the   link   is   valid   for   300
seconds   from   when   this   line   of   code   starts   running
.
$ signedUrl  = $ ossClient -> signUrl ($ bucket , $ object , $
timeout ); # Indicates   the   function   used   to   implement
the   signed   URL .
$ img = $ signedUrl ;
# Indicates   dynamicall  y   placing   the   signed   URL   in
image   resources   and   printing   it   out .
$ my_html  = "< html >";
$ my_html  .= "< img   src =\"".$ img . "\" />";
$ my_html  .= "< p >".$ img ."</ p >";
$ my_html  .= "</ html >";
 echo  $ my_html ;
? >
```

3. If the browser requests the resource multiple times, different signed URLs may be displayed. This is a normal phenomenon because the signed URL changes once it expires. After expiration time the link is no longer valid. It is displayed in Unix time format, for example, Expires=1448991693. The time can be converted to the local time. In Linux, the command for converting the time is `date  - d @ 1448991693` . You can also find a conversion tool on the Internet.

Special instructions

Signed URLs can be used with the Referer whitelist function.

If the expiration time of signed URLs is limited to minutes, even when a leecher spoofs a Referer, the leecher needs to obtain the signed URL and complete leeching before the signed URL expires. Compared with the Referer method, this makes leeching more difficult. Using signed URLs with the Referer whitelist function provides enhanced anti-leech protection results.

Conclusion

Best practices of OSS-based anti-leech protection:

- Use third-level domain name URLs, such as `referer - test . oss - cn - hangzhou . aliyuncs . com / aliyun - logo . png` , as they are more secure than bound second-level domain names. The third-level domain name access method provides bucket-level cleaning and isolation, enabling you to respond to a burst in leeching traffic while preventing different buckets from affecting each other, thereby increasing service availability.

- If you use custom domain names as links, bind the CNAME to a third-level domain name, with the rule bucket + endpoint. For example, your bucket is named "test" and the third-level domain name is `test . oss - cn - hangzhou . aliyuncs . com`.
- Set the strictest possible permission for the bucket. For example, set a bucket that provides Internet services to public-read or private. Do not set it to public-read-write. For bucket permission information, see Access control.
- Verify access sources and set a Referer whitelist based on your requirement.
- If you need a more rigorous anti-leeching solution, consider using signed URLs.
- Record access logs of the bucket, so that you can promptly discover leeching and verify the effectiveness of your anti-leeching solution.

FAQ

- I have configured anti-leech protection on the OSS Console, but the configuration does not take effect. Access to webpages is blocked, whereas access to players is not. Why? How can this problem be fixed?

  Currently, anti-leech protection fails to take effect for audio and video files. When a media player, such as Windows Media Player or Flash Player, is used to request OSS resources, a blank Referer request is sent. This causes anti-leech protection ineffective. To resolve this issue, you can see the preceding signed URL anti-leech protection method.
- What is a Referer? How is it sent? How to deal with HTTPS websites? Does anything else need to be added, like commas?

  A Referer is a request header in the HTTP protocol. It is attached to a request that involves a page jump. You must check whether the Referer in the request sent by your browser is `http ://` or `https ://`. In normal cases, the Referer is `http ://`.

- How are signed URLs generated? Is storing the AccessKeySecret on the client secure?

  See the individual SDK documentation for the method of signing the URL. It is not recommended that the AccessKeySecret be directly stored on the client. RAM provides the STS service to solve this problem. Also, see RAM and STS Guide.
- How do I use wildcard characters (*, ?) to write `a . baidu . com` and `b . baidu . com` ?

  You can use `http ://*. baidu . com` . If the wildcard character represents a single character only, you can also use `http ://?. baidu . com` .
- *.domain.com can match a second-level domain name, but does not match domain.com. Only adding a second entry of domain.com does not work either. What settings must be configured?

  Note that a Referer generally includes a parameter such as http. You can view the request Referer in Chrome’ s developer mode and then specify the corresponding Referer. As in this case, you may have forgotten to include `http ://`, which is required to be `http :// domain . com` .
- What must I do if anti-leech protection does not take effect?

  We recommend that you use Chrome to solve the problem. Open developer mode and click on the Web page to view the `Referer` specific values in the HTTP request. Check whether the `Referer` value matches the Referer value configured on OSS. If they do not match, set the Referer value configured on OSS to the Referer value in the HTTP request. If the problem persists, open a ticket.

6.3 Static website hosting

This document describes the process and procedure about how to build a simple static website based on OSS right from the beginning and also includes FAQs as well. The following are the key steps:

1. Apply for a domain name.
2. Activate OSS and create a bucket.
3. Activate Static Website Hosting on OSS.
4. Access OSS with custom domain names.

Static website hosting overview

You can build a simple static website page based on OSS. Once you activate this function, OSS provides a default homepage and a default 404 page. For more information, see Static Website Hosting in the developer guide.

Procedure

1. Apply for a domain name
2. Activate OSS and create a bucket

   a. Log on to the OSS console and create a bucket named “imgleo23” in Shanghai with the endpoint `oss - cn - shanghai . aliyuncs . com`. For detailed operation, see Create a bucket.

   b. Set the bucket permission to public-read. For detailed operation, see Set bucket ACL.

   c. Upload the content of index.htm and error.htm. For detailed operation, see Upload objects.

   - Body of index.html:

```
< html >
  < head >
      < title > Hello   OSS ! </ title >
      < meta   charset =" utf - 8 ">
  </ head >
  < body >
      < p > Welcome   to   OSS   Static   Website   Hosting .</
 p >
      < p > This   is   the   homepage .</ p >
  </ body >
</ html >
```

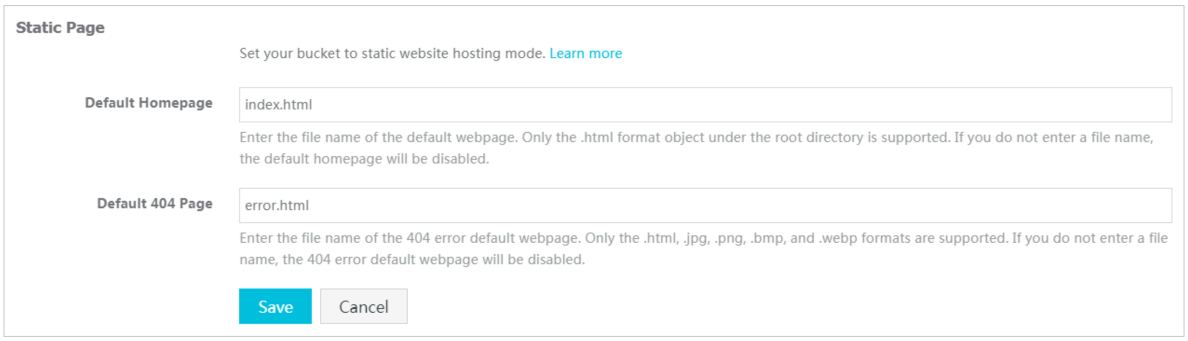
   - Body of error.html:

```
< html >
  < head >
      < title > Hello   OSS ! </ title >
      < meta   charset =" utf - 8 ">
  </ head >
  < body >
      < p > This   is   an   error   homepage   for   OSS
 Static   Website   Hosting .</ p >
  </ body >
</ html >
```

   - `aliyun - logo . png` is a picture.

3. Activate static website hosting on OSS

   As shown in the following figure, once you log on to the OSS console, set `Default Homepage` to `index.html` and `Default 404 Page` to `error.html`. For more information, see Set static website hosting.



Static Page

Set your bucket to static website hosting mode. Learn more

Default Homepage: index.html

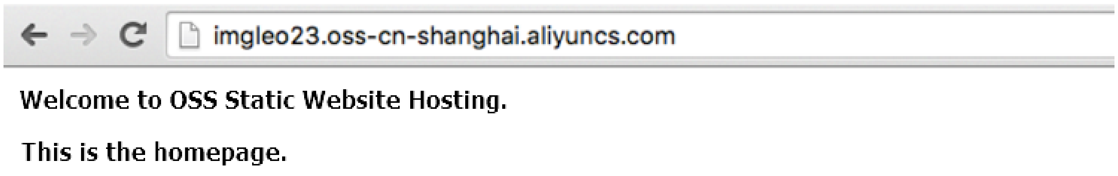
Enter the file name of the default webpage. Only the .html format object under the root directory is supported. If you do not enter a file name, the default homepage will be disabled.

Default 404 Page: error.html

Enter the file name of the 404 error default webpage. Only the .html, .jpg, .png, .bmp, and .webp formats are supported. If you do not enter a file name, the 404 error default webpage will be disabled.

   To test the Static Website Hosting function, enter the URL as shown in the following figure:

   - Display the default homepage:

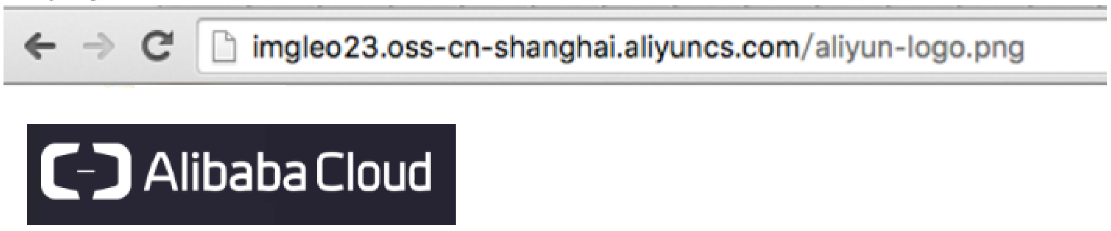


imgleo23.oss-cn-shanghai.aliyuncs.com

Welcome to OSS Static Website Hosting.

This is the homepage.

     When a similar URL is entered, the body of index.html specified upon activating the function is displayed.
   - Display normal files



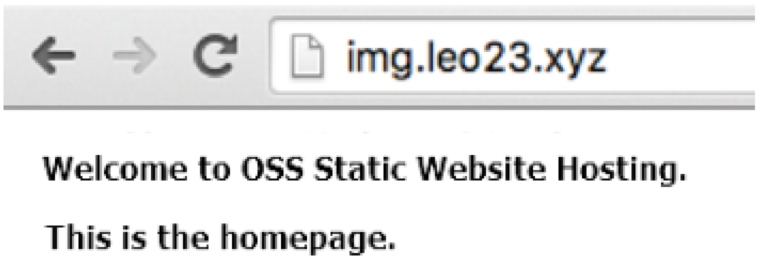
imgleo23.oss-cn-shanghai.aliyuncs.com/aliyun-logo.png

     When a matched file for the entered URL is found, data is read successfully.

4. Access OSS with custom domain names

   For more information about how to access OSS with custom domain names, see Access OSS with custom domain names.

   - Display the default homepage

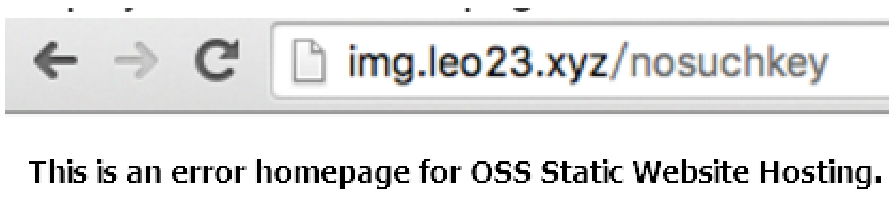
     

     img.leo23.xyz

     Welcome to OSS Static Website Hosting.

     This is the homepage.

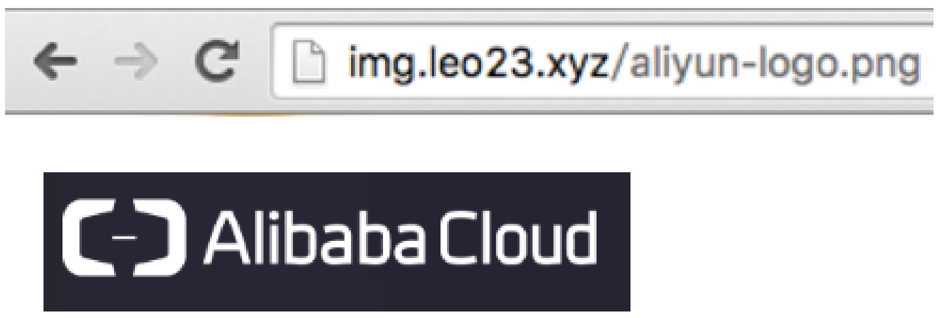
   - Display the default 404 page

     img.leo23.xyz/nosuchkey

     This is an error homepage for OSS Static Website Hosting.

   - Display normal files

     img.leo23.xyz/aliyun-logo.png

Note:
When you use an OSS endpoint in Mainland China regions or the China (Hong Kong) region to access a web file through the Internet , the Content-Disposition: 'attachment=filename;' is automatically added to the Response Header, and the web file is downloaded as an attachment. If you access OSS with a user domain, the Content-Disposition: 'attachment=filename;' will not be added to the Response Header. For more information about using the user domain to access OSS, see Bind a custom domain name.

FAQ

- What are the benefits of OSS Static Website Hosting?

  An ECS instance is saved in case any user needs a relatively small amount of traffic. In the case of larger traffic volumes, CDN can be used.
- How is OSS priced? How does OSS work with CDN?

  For pricing, see the OSS and CDN prices on Alibaba Cloud website. For cases on combination of OSS and CDN, seeCDN-based OSS acceleration practices.
- Do the default homepage and default 404 page both need to be set?

  The default homepage needs to be set, whereas the default 404 page does not need to be set.
- Why does the browser return a 403 error after a URL is entered?

  The reason may be that the bucket permission is not public-read, or your Static Website Hosting function is suspended due to overdue payment.

7 Data security

7.1 Check data transmission integrity by using 64-bit CRC

Background

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

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

Operation instructions

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

  Note:
  The 64-bit CRC value of the entire object is returned for the range get object.
- For copy related operations, for example, Copy Object/Upload Part Copy, the newly generated object/Part may not necessarily have the 64-bit CRC value.

Python example

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

1. Calculate the 64-bit CRC value.

```
import   oss2
from   oss2 . models   import   PartInfo
import   os
import   crcmod
import   random
import   string
do_crc64  =  crcmod . mkCrcFun ( 0x142F0E1E  BA9EA3693L ,  initCrc
= 0L ,  xorOut = 0xffffffff  ffffffffL ,  rev = True )
def   check_crc6  4 ( local_crc6  4 ,  oss_crc64 ,  msg =" check
crc64 "):
if   local_crc6  4  ! =  oss_crc64 :
print  "{ 0 }  check   crc64   failed .  local :{ 1 },  oss :{ 2
}.". format ( msg ,  local_crc6  4 ,  oss_crc64 )
return   False
else :
print  "{ 0 }  check   crc64   ok .". format ( msg )
return   True
def   random_str  ing ( length ):
return  ''. join ( random . choice ( string . lowercase )  for   i
  in   range ( length ))
bucket  =  oss2 .  Bucket ( oss2 .  Auth ( access_key  _id ,
access_key  _secret ),  endpoint ,  bucket_nam  e )
```

2. Verify Put Object.

```
content  =  random_str  ing ( 1024 )
 key  = ' normal - key '
 result  =  bucket . put_object ( key ,  content )
 oss_crc64  =  result . headers . get (' x - oss - hash - crc64ecma
', '')
 local_crc6  4  =  str ( do_crc64 ( content ))
 check_crc6  4 ( local_crc6  4 ,  oss_crc64 , " put   object ")
```

3. Verify Get Object.

```
result  =  bucket . get_object ( key )
 oss_crc64  =  result . headers . get (' x - oss - hash - crc64ecma
', '')
 local_crc6  4  =  str ( do_crc64 ( result . resp . read ()))
 check_crc6  4 ( local_crc6  4 ,  oss_crc64 , " get   object ")
```

4. Verify Upload Part and Complete.

```
part_info_  list  = []
 key  = " multipart - key "
 result  =  bucket . init_multi  part_uploa  d ( key )
 upload_id  =  result . upload_id
 part_1  =  random_str  ing ( 1024  *  1024 )
 result  =  bucket . upload_par  t ( key ,  upload_id ,  1 ,
part_1 )
 oss_crc64  =  result . headers . get (' x - oss - hash - crc64ecma
', '')
 local_crc6  4  =  str ( do_crc64 ( part_1 ))
```

```
# Check   whether   the   uploaded   part   1   data   is
complete
 check_crc6  4 ( local_crc6  4 ,  oss_crc64 , " upload_par  t
object   1 ")
 part_info_  list . append ( PartInfo ( 1 ,  result . etag ,  len (
part_1 )))
 part_2  =  random_str  ing ( 1024  *  1024 )
 result  =  bucket . upload_par  t ( key ,  upload_id ,  2 ,
part_2 )
 oss_crc64  =  result . headers . get (' x - oss - hash - crc64ecma
', '')
 local_crc6  4  =  str ( do_crc64 ( part_2 ))
# Check   whether   the   uploaded   part   2   data   is
complete
 check_crc6  4 ( local_crc6  4 ,  oss_crc64 , " upload_par  t
object   2 ")
 part_info_  list . append ( PartInfo ( 2 ,  result . etag ,  len (
part_2 )))
 result  =  bucket . complete_m  ultipart_u  pload ( key ,
upload_id ,  part_info_  list )
 oss_crc64  =  result . headers . get (' x - oss - hash - crc64ecma
', '')
 local_crc6  4  =  str ( do_crc64 ( part_2 ,  do_crc64 ( part_1 )))
# Check   whether   the   final   object   on   the   OSS   is
consistent   with   the   local   file
 check_crc6  4 ( local_crc6  4 ,  oss_crc64 , " complete   object
")
```

OSS SDK support

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

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

7.2 Protect data through client encryption

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

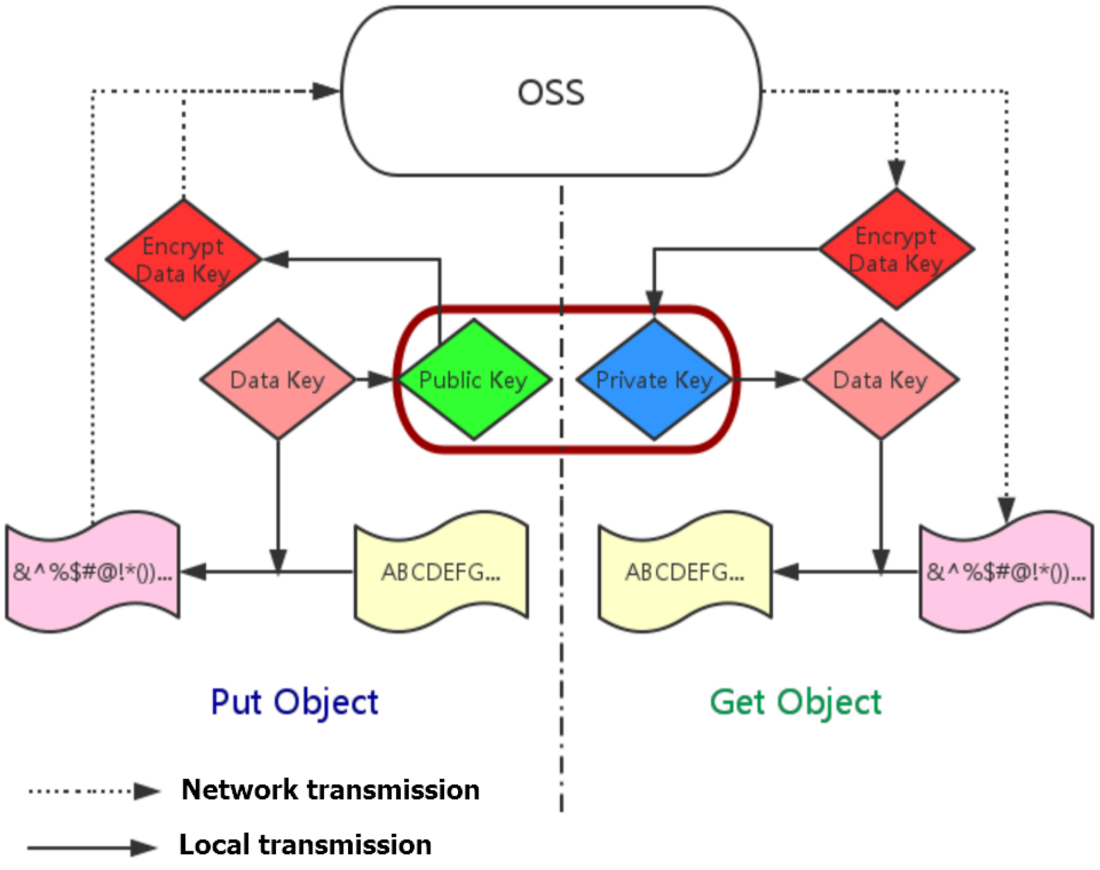
This document describes how to protect data through client encryption based on the current Python SDK version of OSS.

Principles

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

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

Structural diagram



Preparation

1. For installation and usage of the Python SDK, see Quick Installation of Python SDK.
2. Install the PyCrypto library.

```
pip   install   pycrypto
```

Example of complete Python code

```
# -*-  coding :  utf - 8  -*-
 import   os
 import   shutil
 import   base64
 import   random
 import   oss2
 from   Crypto . Cipher   import   PKCS1_OAEP
 from   Crypto . PublicKey   import   RSA
 from   Crypto . Cipher   import   AES
 from   Crypto   import   Random
 from   Crypto . Util   import   Counter
#  aes   256 ,  key   always   is   32   bytes
 _AES_256_K  EY_SIZE  =  32
 _AES_CTR_C  OUNTER_BIT  S_LEN  =  8  *  16
 class   AESCipher :
     def   __init__ ( self ,  key = None ,  start = None ):
         self . key  =  key
```

```
        self . start  =  start
        if  not  self . key :
            self . key  =  Random . new (). read ( _AES_256_K
 EY_SIZE )
        if  not  self . start :
            self . start  =  random . randint ( 1 ,  10 )
        ctr  =  Counter . new ( _AES_CTR_C  OUNTER_BIT  S_LEN ,
 initial_va  lue = self . start )
        self . cipher  =  AES . new ( self . key ,  AES . MODE_CTR ,
 counter = ctr )
    def  encrypt ( self ,  raw ):
        return  self . cipher . encrypt ( raw )
    def  decrypt ( self ,  enc ):
        return  self . cipher . decrypt ( enc )
#  First ,  initialize  the  informatio  n  such  as  AccessKeyI
 d ,  AccessKeyS  ecret ,  and  Endpoint .
#  Obtain  the  informatio  n  through  environmen  t  variables
  or  replace  the  informatio  n  such  as  "< Your
 AccessKeyI  d >"  with  the  real  AccessKeyI  d ,  and  so  on
 .

#  Use  Hangzhou  region  as  an  example .  Endpoint  can  be
 :
#  http :// oss - cn - hangzhou . aliyuncs . com
#  https :// oss - cn - hangzhou . aliyuncs . com
#  Access  using  the  HTTP  and  HTTPS  protocols  respective
 ly .
 access_key  _id  =  os . getenv (' OSS_TEST_A  CCESS_KEY_  ID ', '<
 your  AccessKeyI  d >')
 access_key  _secret  =  os . getenv (' OSS_TEST_A  CCESS_KEY_  SECRET
 ', '< Your  AccessKeyS  ecret >')
 bucket_nam  e  =  os . getenv (' OSS_TEST_B  UCKET ', '< Your
 Bucket >')
 endpoint  =  os . getenv (' OSS_TEST_E  NDPOINT ', '< Your  Access
 Domain  Name >')
#  Make  sure  that  all  the  preceding  parameters  have
 been  filled  in  correctly .
 for  param  in  ( access_key  _id ,  access_key  _secret ,
 bucket_nam  e ,  endpoint ):
    assert  '<'  not  in  param , ' Please  set  the  parameter
 :' +  param
#####  0  prepare  ########
#  0 . 1  Generate  the  RSA  key  file  and  save  it  to
 the  disk
 rsa_privat  e_key_obj  =  RSA . generate ( 2048 )
 rsa_public  _key_obj  =  rsa_privat  e_key_obj . publickey ()
 encrypt_ob  j  =  PKCS1_OAEP . new ( rsa_public  _key_obj )
 decrypt_ob  j  =  PKCS1_OAEP . new ( rsa_privat  e_key_obj )
#  save  to  local  disk
 file_out  =  open (" private_ke  y . pem ", " w ")
 file_out . write ( rsa_privat  e_key_obj . exportKey ())
 file_out . close ()
 file_out  =  open (" public_key . pem ", " w ")
 file_out . write ( rsa_public  _key_obj . exportKey ())
 file_out . close ()
#  0 . 2  Create  the  Bucket  object .  All  the  object -
 related  interfaces  can  be  implemente  d  by  using  the
 Bucket  object
 bucket  =  oss2 .  Bucket ( oss2 .  Auth ( access_key  _id ,
 access_key  _secret ),  endpoint ,  bucket_nam  e )
 obj_name  = ' test - sig - 1 '
 content  = " test  content "
####  1  Put  Object  ####
```

```
#  1 . 1   Generate   the   one - time   symmetric   key   encrypt_ci
 pher   used   to   encrypt   this   object ,  where   key   and
 start   are   values   generated   at   random
 encrypt_ci  pher  =  AESCipher ()
#  1 . 2   Use   the   public   key   to   encrypt   the   informatio
 n   for   assisting   encryption ,  and   save   it   in   the
 custom   meta   of   the   object .  When   Get   Object   is
 performed   later ,  we   can   use   the   private   key   to
 perform   decryption   and   obtain   the   original   content
 according   to   the   custom   meta
 headers  = {}
 headers [' x - oss - meta - x - oss - key '] =  base64 . b64encode (
 encrypt_ob  j . encrypt ( encrypt_ci  pher . key ))
 headers [' x - oss - meta - x - oss - start '] =  base64 . b64encode
 ( encrypt_ob  j . encrypt ( str ( encrypt_ci  pher . start )))
#  1 . 3 .  Use   encrypt_ci  pher   to   encrypt   the   original
 content   to   obtain   encrypt_co  ntent
 encryt_con  tent  =  encrypt_ci  pher . encrypt ( content )
#  1 . 4   Upload   the   object
 result  =  bucket . put_object ( obj_name ,  encryt_con  tent ,
 headers )
 if   result . status  /  100  ! =  2 :
     exit ( 1 )
####  2   Get   Object  ####
#  2 . 1   Download   the   encrypted   object
 result  =  bucket . get_object ( obj_name )
 if   result . status  /  100  ! =  2 :
     exit ( 1 )
 resp  =  result . resp
 download_e  ncrypt_con  tent  =  resp . read ()
#  2 . 2   Resolve   from   the   custom   meta   the   key   and
 start   that   are   previously   used   to   encrypt   this   object

 download_e  ncrypt_key  =  base64 . b64decode ( resp . headers . get
 (' x - oss - meta - x - oss - key ', ''))
 key  =  decrypt_ob  j . decrypt ( download_e  ncrypt_key )
 download_e  ncrypt_sta  rt  =  base64 . b64decode ( resp . headers .
 get (' x - oss - meta - x - oss - start ', ''))
 start  =  int ( decrypt_ob  j . decrypt ( download_e  ncrypt_sta  rt
 ))
#  2 . 3   Generate   the   cipher   used   for   decryption ,  and
 decrypt   it   to   obtain   the   original   content
 decrypt_ci  pher  =  AESCipher ( key ,  start )
 download_c  ontent  =  decrypt_ci  pher . decrypt ( download_e
 ncrypt_con  tent )
 if   download_c  ontent  ! =  content :
     print  " Error !"
 else :
     print  " Decrypt   ok .  Content   is : % s " %  download_c
 ontent
```

7.3 Protect data by performing server-side encryption

You can protect static data by performing server-side encryption. If you enable the server-side encryption function, OSS encrypts user data (that is, the objects) when writing the data into the hard disks deployed in the data center and automatically decrypts the data when it is accessed. Authentication is performed on users who access the encrypted data.

Note:
For more information about server-side encryption, see Server-side encryption.

OSS supports the following three server-side encryption methods:

- Server-side encryption fully managed by OSS (SSE-OSS)

  When sending a request to upload an object or modify the metadata of an object, you can include the `X - OSS - server - side - encryption` header in the request and specify its value as AES256. In this method, OSS uses AES256 to encrypt each object with an individual key. Furthermore, the individual keys are encrypted by a customer master key (CMK) that is updated periodically for higher security.
- Server-side encryption using the default managed CMK (SSE-KMS)

  When sending a request to upload an object or modify the metadata of an object, you can include the `X - OSS - server - side - encryption` header in the request and specify its value as KMS without a specified CMK ID. In this method, OSS generates an individual key to encrypt each object by using the default managed CMK, and automatically decrypts the object when it is downloaded.
- Server-side encryption using a CMK specified by the user (SSE-KMS)

  When sending a request to upload an object or modify the metadata of an object, you can include the `X - OSS - server - side - encryption` header in the request, specify its value as KMS, and specify the value of `X - oss - server - side - encryption - key - id` to a specified CMK ID. In this method. OSS generates an individual key to encrypt each object by using the specified CMK, and adds the CMK ID used to encrypt an object into the metadata of the object so that the object is automatically decrypted when it is downloaded by an authorized user. You can use key material generated by the system automatically or import key material from an external source.

  Notice:

  - The server-side encryption method using a specified CMK is in the beta testing phase. To use the method, contact Alibaba Cloud technical support.

- Only one server-side encryption method can be used for an object at one time.
- If you use a CMK to encrypt an object, the data key used in the encryption is also encrypted and is stored as the metadata of the object.
- In server-side encryption that uses the default managed CMK, only the data in the object is encrypted. The metadata of the object is not encrypted.
- Fees for API calls are incurred if you use a CMK to encrypt an object.
- To use a RAM user to encrypt objects with a specified CMK, you must grant the relevant permissions to the RAM user. For more information, see Use RAM for KMS resource authorization.

Perform server-side encryption fully managed by OSS

1. Log on to the OSS console and create a bucket. For more information, see Create a bucket.
2. Upload an object in plaintext to OSS. For more information, see Upload an object.
3. Encrypt the uploaded object by running the following Python script:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all the APIs in
 OSS. We recommend that you log on as a RAM user to access APIs
 or perform routine operations and maintenance. To create a RAM
 account, log on to the  RAM  console .
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>
')
# This example uses the endpoint oss-cn-hongkong. Specify the actual
 endpoint based on your requirements.

bucket = oss2.Bucket(auth, 'http://oss-cn-hongkong.aliyuncs.com',
 'test-hongkong-2025')

bucket.update_object_meta('01.txt',{'x-oss-server-side-
encryption':'AES256'})
```

4. Verify the encryption result.

   Use ossutil to view the object before and after the encryption.

   - Before encryption:

```
D :\ 5 - AK_account \ ossutil64 > ossutil64 . exe   stat    oss
:// test - hongkong - 2025 / 01 . txt
ACL                           :  default
Accept - Ranges                 :  bytes
Content - Length                :  62
Content - Md5                   :  k2GA4LeqHv  VpQvBfnleN  Og ==
Content - Type                  :  text / plain
Etag                          :  936180E0B7  AA1EF56942
F05F9E578D  3A
```

```
Last - Modified                 :  2018 - 10 - 2420 : 41 : 54  +
0800   CST
Owner                         :  14166xxxxx  x36597
X - Oss - Hash - Crc64ecma          :  9888192182  077127097
X - Oss - Object - Type             :  Normal
X - Oss - Storage - Class           :  Standard
```

- **After encryption:**

```
D :\ 5 - AK_account \ ossutil64 > ossutil64 . exe   stat    oss
:// test - hongkong - 2025 / 01 . txt
ACL                           :  default
Accept - Ranges                 :  bytes
Content - Length                :  62
Content - Md5                   :  k2GA4LeqHv  VpQvBfnleN  Og ==
Content - Type                  :  text / plain
Etag                          :  936180E0B7  AA1EF56942
F05F9E578D  3A
Last - Modified                 :  2018 - 10 - 2420 : 46 : 39  +
0800   CST
Owner                         :  14166xxxxx  x36597
X - Oss - Hash - Crc64ecma          :  9888192182  077127097
X - Oss - Object - Type             :  Normal
X - Oss - Server - Side - Encryption :  AES256
X - Oss - Storage - Class           :  Standard
```

Perform server-side encryption using the default CMK managed by OSS

1. Log on to the OSS console and create a bucket. For more information, see Create a bucket.
2. Upload an object in plaintext to OSS. For more information, see Upload an object.
3. Activate KMS in Alibaba Cloud product management page.
4. Encrypt the uploaded object by running the following Python script:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all the APIs in
 OSS. We recommend that you log on as a RAM user to access APIs
 or perform routine operations and maintenance. To create a RAM
 account, log on to the  RAM   console .
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>
')
This example uses the endpoint oss-cn-hongkong. Specify the actual
 endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hongkong.aliyuncs.com',
 'test-hongkong-2025')
```

```
bucket.update_object_meta('01.txt',{'x-oss-server-side-
encryption':'KMS'})
```

Perform server-side encryption using a CMK specified by the user

1. Log on to the OSS console and create a bucket. For more information, see Create a bucket.
2. Upload an object in plaintext to OSS. For more information, see Upload an object.
3. Activate KMS in Alibaba Cloud product management page.
4. Log on to the KMS console. Click Create Key and configure the following options to create a CMK in the same region as the OSS bucket.
   - Customize the description for the key in Description.
   - Select Alibaba Cloud KMS for Key Material Source under Advanced.

   Note:
   You can also import an external key. For more information, see Import key material.
5. Use the ID of the created CMK to encrypt the upload object by running the following Python script:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all the APIs in
 OSS. We recommend that you log on as a RAM user to access APIs
 or perform routine operations and maintenance. To create a RAM
 account, log on to the  RAM  console .
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>
')
# This example uses the endpoint oss-cn-hongkong. Specify the actual
 endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hongkong.aliyuncs.com',
 'test-hongkong-2025')

bucket.update_object_meta('01.txt',{'x-oss-server-side-
encryption':'KMS','x-oss-server-side-encryption-key-id':
 '33701a45-6723-4a04-a367-68c060382652'})
```

8 OSS resource monitoring and alarm service

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

Prerequisites

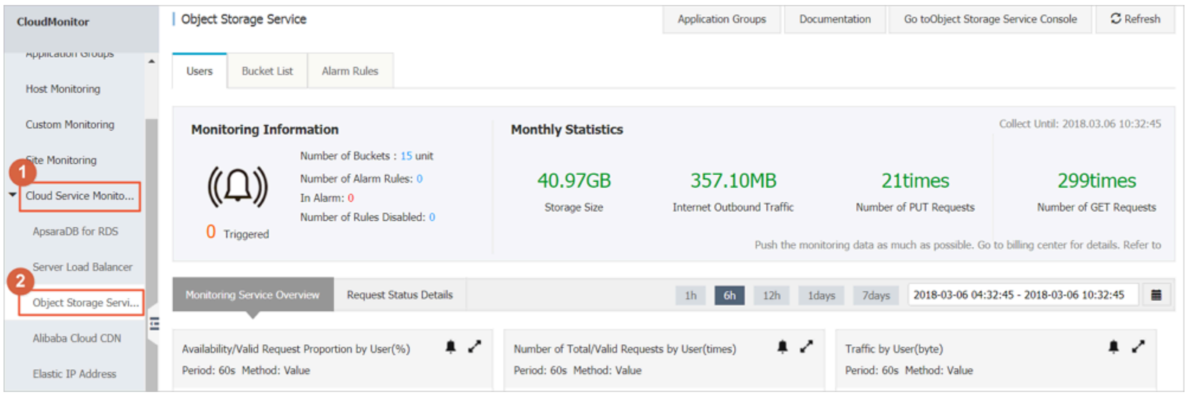
- Activate the OSS service.
- Activate the CloudMonitor service.

Monitor OSS resources

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

   You can obtain monitoring data on the OSS monitoring page.

   Note:
   "by User" refers to user-level data, that is, all bucket data of this user.



Monitoring Information

Number of Buckets : 15 unit
Number of Alarm Rules: 0
In Alarm: 0
Number of Rules Disabled: 0
0 Triggered

Monthly Statistics

Collect Until: 2018.03.06 10:32:45

40.97GB	357.10MB	21times	299times
Storage Size	Internet Outbound Traffic	Number of PUT Requests	Number of GET Requests

Push the monitoring data as much as possible. Go to billing center for details. Refer to

2018-03-06 04:32:45 - 2018-03-06 10:32:45

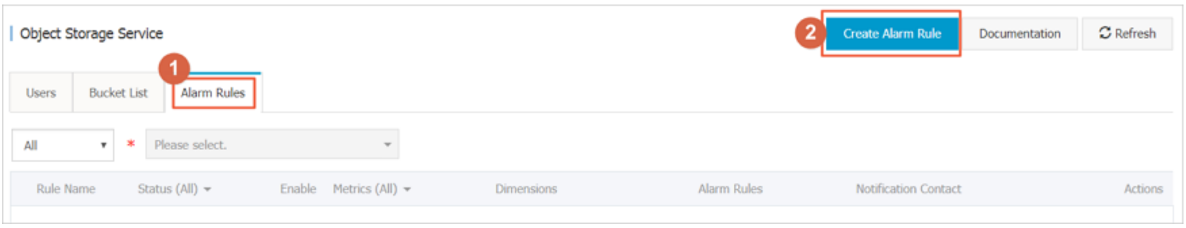
Availability/Valid Request Proportion by User(%)
Period: 60s Method: Value

Number of Total/Valid Requests by User(times)
Period: 60s Method: Value

Traffic by User(byte)
Period: 60s Method: Value

Set alarm rules

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



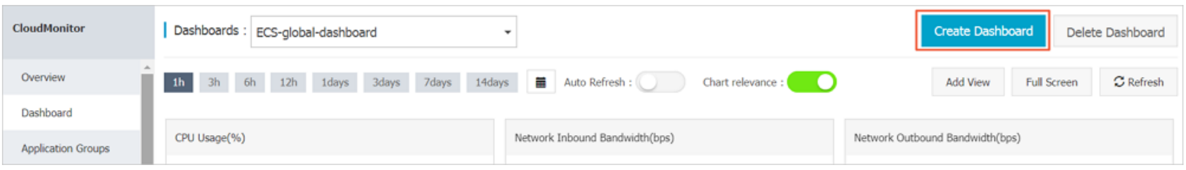
2. Configure your alarm rules.

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

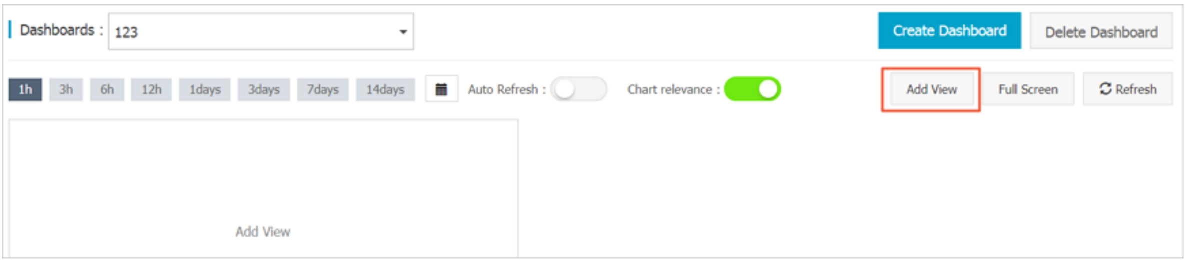
Custom monitoring dashboard

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

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



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



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

   For configuration details, see Monitoring indicators reference.

9 OSS performance and scalability best practice

Partitions and naming conventions

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

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

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

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

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

- Example 1: Add hex hash prefixes into file names

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

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

```
...
```

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

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

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

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

- Example 2: Reverse the file name

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

```
 sample - bucket - 02 / 1513160001  245 . log
 sample - bucket - 02 / 1513160001  722 . log
 sample - bucket - 02 / 1513160001  836 . log
 sample - bucket - 02 / 1513160001  956 . log
...
 sample - bucket - 02 / 1513160002  153 . log
 sample - bucket - 02 / 1513160002  556 . log
 sample - bucket - 02 / 1513160002  859 . log
...
```

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

```
 sample - bucket - 02 / 5421000613  151 . log
 sample - bucket - 02 / 2271000613  151 . log
 sample - bucket - 02 / 6381000613  151 . log
 sample - bucket - 02 / 6591000613  151 . log
...
 sample - bucket - 02 / 3512000613  151 . log
 sample - bucket - 02 / 6552000613  151 . log
 sample - bucket - 02 / 9582000613  151 . log
...
```

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

10 Terraform

10.1 Introduction

Terraform is an open-source automatic resource orchestration tool that supports multiple cloud service providers. Alibaba Cloud (referenced as terraform-alicloud-provider in Terraform) allows developers to easily build, update, and version their infrastructure in the Alibaba Cloud Terraform ecosystem by supporting over 90 resources and data sources across more than 20 products and services.

HashiCorp Terraform is an automatic IT infrastructure orchestration tool that can manage and maintain IT resources by using code. The easy to use Command Line Interface (CLI) of Terraform allows you to deploy configuration files on Alibaba Cloud or any other supported cloud, and control the versions of the configuration files. The CLI provides code for the infrastructures (such as VMs, storage accounts, and network interfaces) defined in the configuration files that describe the cloud resource topology. The Command Line Interface (CLI) of Terraform provides a simple mechanism, which is used to deploy configuration files on Alibaba Cloud or any other supported cloud and control the versions of the configuration files. Terraform is a highly scalable tool that supports new infrastructures through providers. You can use Terraform to create, modify, or delete multiple resources, such as ECS, VPC, RDS, and SLB.

Functions of OSS Terraform module

You can use the OSS Terraform module to manage buckets and objects. For example:

- Bucket management functions:
  - Creates a bucket.
  - Configures an ACL for a bucket.
  - Configures Cross-Origin Resource Sharing (CORS) for a bucket.
  - Sets logging for a bucket.
  - Configures static website hosting for a bucket.
  - Configures referers for a bucket.
  - Configures the lifecycle rules of a bucket.

- Object management functions:
  - Uploads an object.
  - Configures server-end encryption for an object.
  - Sets an ACL for an object.
  - Sets Object Meta.

References

- For the installation and usage of Terraform, see Use Terraform to manage OSS.
- To download the OSS Terraform module, see terraform-alicloud-modules.
- For more information about the OSS Terraform module, see alicloud_oss_bucket.

10.2 Use Terraform to manage OSS

This topic describes how to install and configure Terraform and how to use Terraform to manage OSS.

Install and configure Terraform

Before using Terraform, follow these steps to install and configure Terraform:

1. Download the software package applicable to your operating system from Terraform official website. In this topic, Terraform is installed and configured in a Linux operating system as an example.
2. Extract the software package to the */ usr / local / bin* directory. If you extract the executable file to another directory, you must add the directory to global variables.
3. Run Terraform to verify the directory configuration. If a list of available Terraform options is displayed, Terraform is installed.

```
[ root @ test   bin ]# terraform
```

```
Usage :  terraform  [- version ] [- help ] < command > [ args ]
```

4. Create and authorize a RAM user.

   a. Log on to the RAM console.
   b. Create a RAM user named *Terraform* and create an AccessKey for the user. For more information, see Create a RAM user.
   c. Authorize the RAM user. You can add relevant permissions to the Terraform RAM user as needed. For detailed steps, see Authorize RAM users.

   Notice:
   To maintain data security, do not use the AccessKey of your Alibaba Cloud account to configure Terraform.

5. Create a test directory named *terraform - test* . You must create a separate directory for each Terraform project.

```
[ root @ test   bin ]# mkdir   terraform - test
```

6. Enter the *terraform - test* directory.

```
[ root @ test   bin ]# cd   terraform - test
[ root @ test   terraform - test ]#
```

7. Create a configuration file. Terraform reads all **. tf* and **. tfvars* files in the *terraform - test* directory when running. Therefore, you can write configuration information to different files as needed. Some common configuration files are described as follows:

```
 provider . tf              --  Used   to   configure   providers .
 terraform . tfvars         --  Used   to   configure   the
 variables   required   to   configure   providers .
 varable . tf               --  Used   to   configure   universal
 variables .
 resource . tf              --  Used   to   define   resources .
 data . tf                  --  Used   to   define   package   files
 .
 output . tf                --  Used   to   configure   the   output
 .
```

For example, when you create the *provider . tf* file, you can configure your authentication information as follows:

```
[ root @ test   terraform - test ]#  vim   provider . tf
 provider  " alicloud " {
     region            = " cn - beijing "
     access_key   = " LTA ********** NO2 "
     secret_key    = " MOk8x0 ******************** wwff "
```

For more information about configurations, see alicloud_oss_bucket.

8. Initialize your working directory.

```
[ root @ test   terraform - test ]# terraform   init


 Initializi  ng   provider   plugins ...
-  Checking   for   available   provider   plugins   on   https ://
 releases . hashicorp . com ...
-  Downloadin  g   plugin   for   provider  " alicloud " ( 1 . 25 .
 0 )...




 The   following   providers   do   not   have   any   version
 constraint  s   in   configurat  ion ,
 so   the   latest   version   was   installed .


 To   prevent   automatic   upgrades   to   new   major   versions
 that   may   contain   breaking
 changes ,  it   is   recommende  d   to   add   version  = "..."
 constraint  s   to   the
 correspond  ing   provider   blocks   in   configurat  ion ,  with
   the   constraint   strings
 suggested   below .


*  provider . alicloud :  version   = "~>  1 . 25 "


 Terraform   has   been   successful  ly   initialize  d !


 You   may   now   begin   working   with   Terraform .  Try
 running  " terraform   plan "  to   see
 any   changes   that   are   required   for   your   infrastruc
 ture .  All   Terraform   commands
 should   now   work .


 If   you   ever   set   or   change   modules   or   backend
 configurat  ion   for   Terraform ,
 rerun   this   command   to   reinitiali  ze   your   working
 directory .  If   you   forget ,  other
 commands   will   detect   it   and   remind   you   to   do   so
 if   necessary .
```

Notice:

After creating a working directory and configuration files for a Terraform project, you must initialize the working directory.

You can use Terraform after completing the preceding steps.

Use Terraform to manage OSS

After Terraform is installed, you can run commands in Terraform to manage OSS. Some common commands are described as follows:

- `terraform plan` : You can run this command to view the operations to be executed by a configuration file.

  For example, you add a configuration file named *test . tf* that is used to create a bucket as follows:

  ```
  [ root @ test   terraform - test ]# vim   test . tf
   resource  " alicloud_o  ss_bucket " " bucket - acl "{
     bucket  = " figo - chen - 2020 "
     acl  = " private "
  }
  ```

  In this case, you can run the `terraform plan` command to view the operations to be executed by the *test . tf* configuration file.

  ```
  [ root @ test   terraform - test ]#  terraform   plan
   Refreshing   Terraform   state   in - memory   prior   to   plan
   ...
   The   refreshed   state   will   be   used   to   calculate   this
     plan ,  but   will   not   be
   persisted   to   local   or   remote   state   storage .


  -------------------------------------------------------------------------

   An   execution   plan   has   been   generated   and   is   shown
   below .
   Resource   actions   are   indicated   with   the   following
   symbols :
    +  create

   Terraform   will   perform   the   following   actions :

    +  alicloud_o  ss_bucket . bucket - acl
         id :                < computed >
         acl :               " private "
         bucket :            " figo - chen - 2020 "
         creation_d  ate :     < computed >
         extranet_e  ndpoint : < computed >
         intranet_e  ndpoint : < computed >
         location :          < computed >
         logging_is  enable :  " true "
         owner :             < computed >
         referer_co  nfig .#:  < computed >
         storage_cl  ass :     < computed >


   Plan :  1   to   add ,  0   to   change ,  0   to   destroy .

  -------------------------------------------------------------------------
  ```

```
 Note :  You   didn ' t   specify   an  "- out "  parameter   to
 save   this   plan ,  so   Terraform
 can ' t   guarantee   that   exactly   these   actions   will   be
   performed   if
" terraform   apply "  is   subsequent  ly   run .
```

- `terraform apply` : You can run this command to execute a configuration file in the working directory.

  For example, if you want to create a bucket named *figo - chen - 2020* , you must add a configuration file named *test . tf* that is used to create the bucket as follows:

```
[ root @ test   terraform - test ]# vim   test . tf
 resource  " alicloud_o  ss_bucket " " bucket - acl "{
   bucket  = " figo - chen - 2020 "
   acl  = " private "
}
```

  Then you can run the `terraform apply` command to execute the configuration file.

```
[ root @ test   terraform - test ]# terraform    apply

 An   execution   plan   has   been   generated   and   is   shown
 below .
 Resource   actions   are   indicated   with   the   following
 symbols :
  +  create

 Terraform   will   perform   the   following   actions :

  +  alicloud_o  ss_bucket . bucket - acl
       id :                 < computed >
       acl :                " private "
       bucket :             " figo - chen - 2020 "
       creation_d  ate :     < computed >
       extranet_e  ndpoint : < computed >
       intranet_e  ndpoint : < computed >
       location :           < computed >
       logging_is  enable :  " true "
       owner :              < computed >
       referer_co  nfig .#:  < computed >
       storage_cl  ass :     < computed >


 Plan :  1   to   add ,  0   to   change ,  0   to   destroy .

 Do   you   want   to   perform   these   actions ?
   Terraform   will   perform   the   actions   described   above .
   Only  ' yes '  will   be   accepted   to   approve .

   Enter   a   value :  yes

 alicloud_o  ss_bucket . bucket - acl :  Creating ...
   acl :                "" => " private "
   bucket :             "" => " figo - chen - 2020 "
   creation_d  ate :     "" => "< computed >"
```

```
    extranet_e  ndpoint : "" => "< computed >"
    intranet_e  ndpoint : "" => "< computed >"
    location :          "" => "< computed >"
    logging_is  enable :  "" => " true "
    owner :             "" => "< computed >"
    referer_co  nfig .#:  "" => "< computed >"
    storage_cl  ass :     "" => "< computed >"
  alicloud_o  ss_bucket . bucket - acl :  Creation   complete   after
    1s   ( ID :  figo - chen - 2020 )

  Apply   complete !  Resources : 1   added , 0   changed , 0
  destroyed .
```

Note:

After you execute the configuration file, a new bucket is created if the *figo - chen - 2020* bucket does not exist. If the *figo - chen - 2020* bucket already exists and is an empty bucket that is created by Terraform, the bucket is deleted and a new bucket with the same name is created.

- `terraform   destroy`: You can run this command to delete an empty bucket that is created by Terraform.
- `terraform   import`: You can run this command to import a bucket that is not created by Terraform.

  For example, you create a configuration file named *main . tf* and add configurations to the file as follows:

  ```
  [ root @ test   terraform - test ]# vim   main . tf
   resource   " alicloud_o  ss_bucket " " bucket " {
    bucket   = " test - hangzhou - 2025 "
    acl   = " private "
  }
  ```

  Then you can run the following command to import the *test - hangzhou - 2025* bucket:

  ```
  terraform   import   alicloud_o  ss_bucket . bucket   test -
  hangzhou - 2025
  ```

References

- For more bucket configuration examples, see alicloud_oss_bucket.
- For more object configuration examples, see alicloud_oss_bucket_object.