Alibaba Cloud
Object Storage Service

Best Practices

Issue: 20181106

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.
5. By law, all the content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade

secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).

6. Please contact Alibaba Cloud directly if you discover any errors in this document.

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 information, supplementary instructions, and other content that the user must understand.	 **Note:** 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]*
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich` *{stand | slave}*

Contents

1 Application server

1.1 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:ListObjects"
          ],
          "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 corresponding to different prefixes are spatially isolated in the bucket.

```
{
    "Statement": [
      {
        "Action": [
          "oss:GetObject",
          "oss:ListObjects"
        ],
        "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
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

Operations on OSS upon retrieving STS token	Result
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:DeleteObject",
          "oss:ListParts",
          "oss:AbortMultipartUpload",
          "oss:ListObjects"
        ],
        "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:DeleteObject",
          "oss:ListParts",
          "oss:AbortMultipartUpload",
          "oss:ListObjects"
        ],
        "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 "AccessKeyId", 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*

2 Migrate data to OSS

2.1 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. Download*emr-tools*.

   Note:
   emr-tools is compatible with Hadoop versions 2.4.x, 2.5.x, 2.6.x, and 2.7.x. If you require compatibility with other Hadoop versions, *open a ticket*.

2. Extract the compressed tool to a local directory.

   ```
   tar jxf emr-tools.tar.bz2
   ```

3. Copy HDFS data to OSS.

   ```
   cd emr-tools
   ```

```
./hdfs2oss4emr.sh /path/on/hdfs oss://accessKeyId:accessKeySecret@
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 AccessKeySecret*.
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_1493800598643_0009
completed successfully
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-milliseconds taken by all map tasks=30010
        Total megabyte-milliseconds 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=17317421056
        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
         BYTESCOPIED=258660
         BYTESEXPECTED=258660
         COPY=10
copy from /path/on/hdfs to oss://accessKeyId:accessKeySecret@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.

```
./hdfs2oss4emr.sh oss://accessKeyId:accessKeySecret@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*.

2.2 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.

Items	Amazon S3 permissions	Amazon S3	Alibaba Cloud OSS
	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.
	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.

Items	Amazon S3 permissions	Amazon S3	Alibaba Cloud OSS
	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.

  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

3 Bucket management

3.1 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:

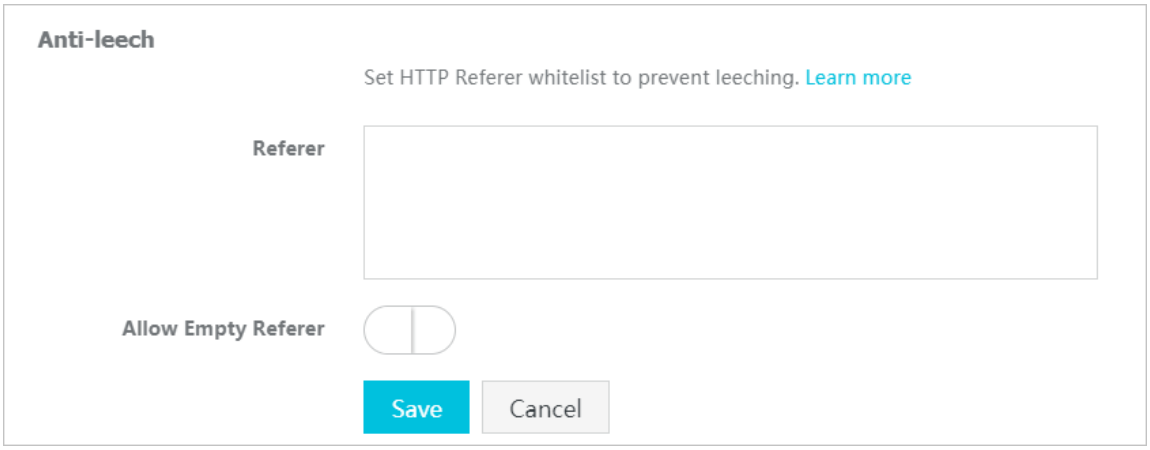
- Asterisks '*': You can use an asterisks instead of 0 or more characters. If you are looking for a file name that starts with “AEW”, you can enter `AEW` to search for all types of files with the names starting with “AEW”, for example, AEWT.txt, AEWU.EXE, and AEWI.dll. If you want to narrow down the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



  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.

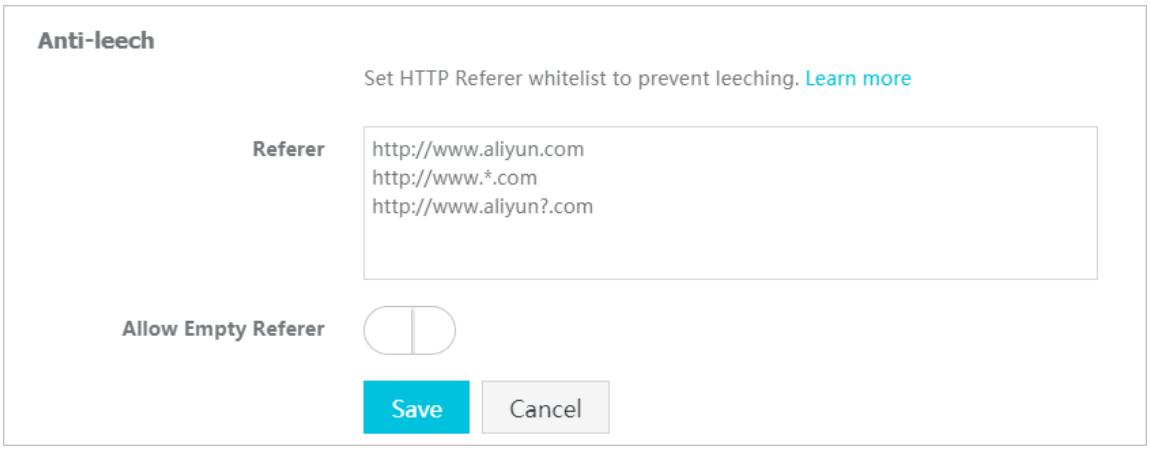
  — Disable Allow Empty Referer and configure a Referer whitelist.

  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:



After testing, the following results are obtained:

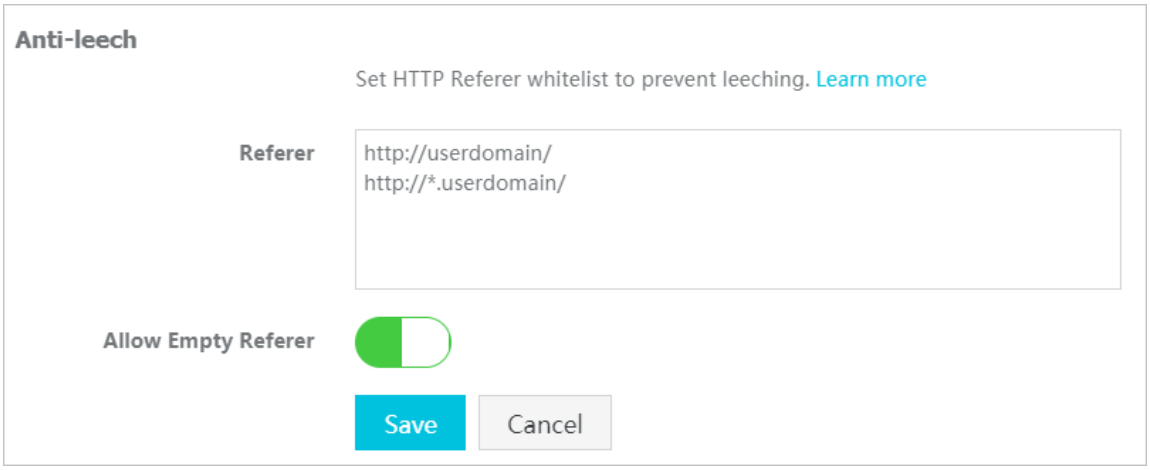
Browser input	Expectation	Result
http://referer-test.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png	Expectation for direct access with a blank Referer: Blank Referers are not allowed and OSS returns 403.	As expected
http://userdomain/error.html	Expectation for a request from the origin site: successful 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:



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
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-read.
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.
 $accessKeyId="a5etodit71tlznjt3pdx7lch";
 #Indicates the AccessKeyId, which must be replaced by the one you
use.
 $accessKeySecret="secret_key";
 #Indicates the AccessKeySecret, 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.
 $ossClient = new OssClient($accessKeyId, $accessKeySecret, $
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 dynamically 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. For access log information, see *Access logging configuration*.

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.

3.2 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:

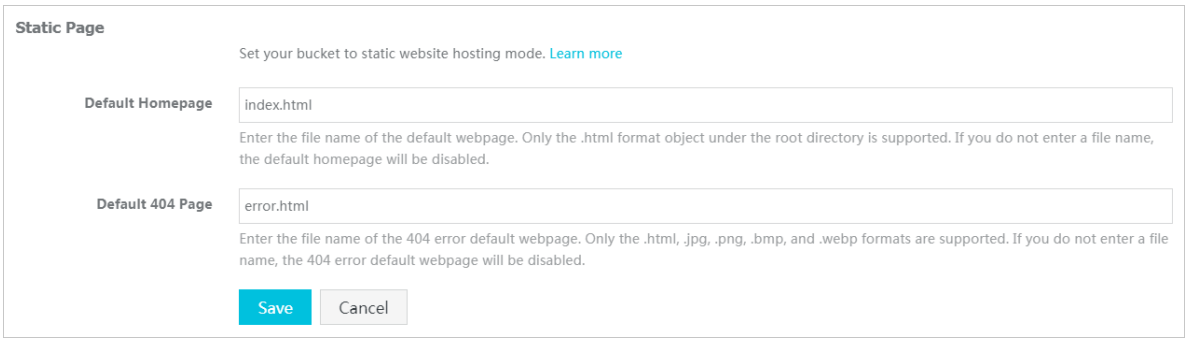
```
<html>
  <head>
      <title>Hello OSS! </title>
      <meta charset="utf-8">
  </head>
```

```
  <body>
      <p>This is an error homepage for OSS Static Website
Hosting.</p>
  </body>
</html>
```

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

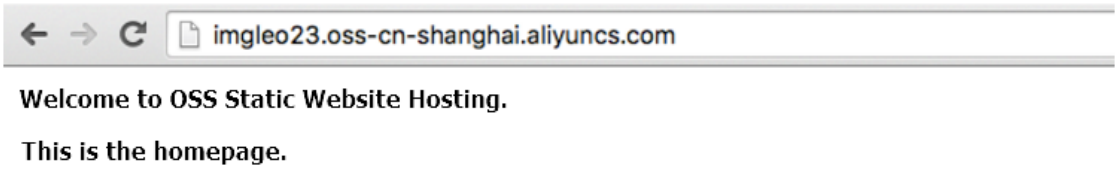
3. Activate static website hosting on OSS

   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*.



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

   - Display the default homepage:



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

   - Display normal files

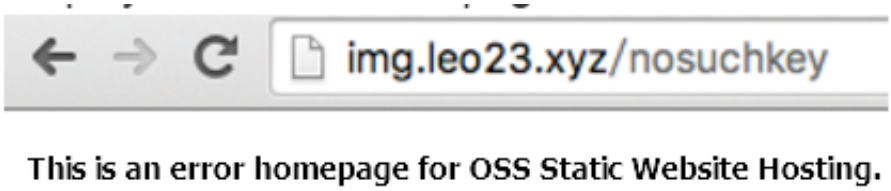
     When a matched file for the entered URL is found, data is read successfully.

4. Access OSS with custom domain names

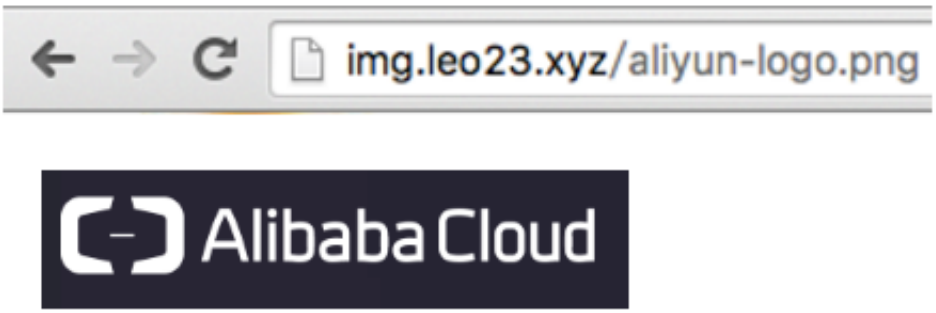
For more information about how to access OSS with custom domain names, see *Access OSS with custom domain names*.

- Display the default homepage

- Display the default 404 page



- Display normal files



Note:

When you use an OSS endpoint in Mainland China regions or the Hongkong 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, see*CDN-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.

4 Access control

4.1 Bucket permission separation

Another scenario is introduced in this section. If another user is using the developed app, you can use an individual bucket to store your app data. Assume that the bucket is the ram-test-app. In consideration of permission separation, the application server must not be allowed to access the ram-test-app; that is, the account ram_test_pub is permitted only to read ram-test-dev. This can also be realized through the RAM permission system. The procedure is as follows:

1. Because the system has no default bucket-level policy, we must create a custom policy.

   The bucket access policy is shown as follows. For more information, see *RAM Policy Description* and *OSS Authorization FAQs* .

   ```
   {
   "Version": "1",
   "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "oss:ListObjects",
        "oss:GetObject"
      ],
      "Resource": [
        "acs:oss:*:*:ram-test-dev",
        "acs:oss:*:*:ram-test-dev/*"
      ]
    }
   ]
   }
   ```

   After setting, we can see the policy in the custom authorization policy list.

2. In user authorization management, add this policy to the selected authorization policy list. Also in **Users** > **Management** > **Authorization policy**, all previously granted OSS read permissions can be revoked.

3. Test the validity of permission configured.

   - The object in ram-test-dev can be accessed:

     ```
     $./osscmd get oss://ram-test-dev/test.txt test.txt --host=oss-cn
     -hangzhou.aliyuncs.com -i oOhue******Frogv -k OmVwFJO3qcT0******
     FhOYpg3p0KnA
     100% The object test.txt is downloaded to test.txt, please check.
     ```

```
0.047(s) elapsed
```

- The object in ram-test-app cannot be accessed:

```
$./osscmd get oss://ram-test-app/test.txt test.txt --host=oss-cn
-hangzhou.aliyuncs.com -i oOhue******Frogv -k OmVwFJO3qcT0******
FhOYpg3p0KnA
 Error Headers:
 [('content-length', '229'), ('server', 'AliyunOSS'), ('connection
', 'keep-alive'), ('x-oss-request-id', '5646ED53F9EEA2F3324191A2
'), ('date', 'Sat, 14 Nov 2015 08:14:11 GMT'), ('content-type', '
application/xml')]
 Error Body:
 <? xml version="1.0" encoding="UTF-8"? >
 <Error>
   <Code>AccessDenied</Code>
   <Message>AccessDenied</Message>
   <RequestId>5646ED53F9EEA2F3324191A2</RequestId>
   <HostId>ram-test-app.oss-cn-hangzhou.aliyuncs.com</HostId>
   </Error>
 Error Status:
 403
 get Failed!
```

- Files cannot be uploaded to oss-test-app:

```
$./osscmd put test.txt oss://ram-test-app/test.txt --host=oss-cn
-hangzhou.aliyuncs.com -i oOhue******Frogv -k OmVwFJO3qcT0******
FhOYpg3p0KnA
 100% Error Headers:
 [('content-length', '229'), ('server', 'AliyunOSS'), ('connection
', 'keep-alive'), ('x-oss-request-id', '5646ED7BB8DE437A912DC7A8
'), ('date', 'Sat, 14 Nov 2015 08:14:51 GMT'), ('content-type', '
application/xml')]
 Error Body:
 <? XML version = "1.0" encoding = "UTF-8 "? >
 <Error>
   <Code>AccessDenied</Code>
   <Message>AccessDenied</Message>
   <RequestId>5646ED7BB8DE437A912DC7A8</RequestId>
   <HostId>ram-test-app.oss-cn-hangzhou.aliyuncs.com</HostId>
 </Error>
 Error status:
 403
 put Failed!
```

Using the preceding configuration, we have successfully separated the permissions for ram-test-dev and ram-test-app.

The preceding section explains how to use the subaccount permission control function to separate permissions and minimize the potential risk of information leakage.

If you want to implement more complex access control, see *RAM User Guide*.

5 Data security

5.1 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(0x142F0E1EBA9EA3693L, initCrc=0L, xorOut=
0xffffffffffffffffL, rev=True)
def check_crc64(local_crc64, oss_crc64, msg="check crc64"):
if local_crc64 ! = oss_crc64:
print "{0} check crc64 failed. local:{1}, oss:{2}.".format(msg,
local_crc64, oss_crc64)
return False
else:
print "{0} check crc64 ok.".format(msg)
return True
def random_string(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_name)
```

2. Verify Put Object.

```
content = random_string(1024)
 key = 'normal-key'
 result = bucket.put_object(key, content)
 oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
 local_crc64 = str(do_crc64(content))
 check_crc64(local_crc64, oss_crc64, "put object")
```

3. Verify Get Object.

```
result = bucket.get_object(key)
 oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
 local_crc64 = str(do_crc64(result.resp.read()))
 check_crc64(local_crc64, oss_crc64, "get object")
```

4. Verify Upload Part and Complete.

```
part_info_list = []
 key = "multipart-key"
 result = bucket.init_multipart_upload(key)
 upload_id = result.upload_id
 part_1 = random_string(1024 * 1024)
 result = bucket.upload_part(key, upload_id, 1, part_1)
 oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
 local_crc64 = str(do_crc64(part_1))
 #Check whether the uploaded part 1 data is complete
 check_crc64(local_crc64, oss_crc64, "upload_part object 1")
 part_info_list.append(PartInfo(1, result.etag, len(part_1)))
 part_2 = random_string(1024 * 1024)
 result = bucket.upload_part(key, upload_id, 2, part_2)
 oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
 local_crc64 = str(do_crc64(part_2))
 #Check whether the uploaded part 2 data is complete
 check_crc64(local_crc64, oss_crc64, "upload_part object 2")
 part_info_list.append(PartInfo(2, result.etag, len(part_2)))
```

```
 result = bucket.complete_multipart_upload(key, upload_id,
part_info_list)
 oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
 local_crc64 = str(do_crc64(part_2, do_crc64(part_1)))
 #Check whether the final object on the OSS is consistent with the
local file
 check_crc64(local_crc64, 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*

5.2 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

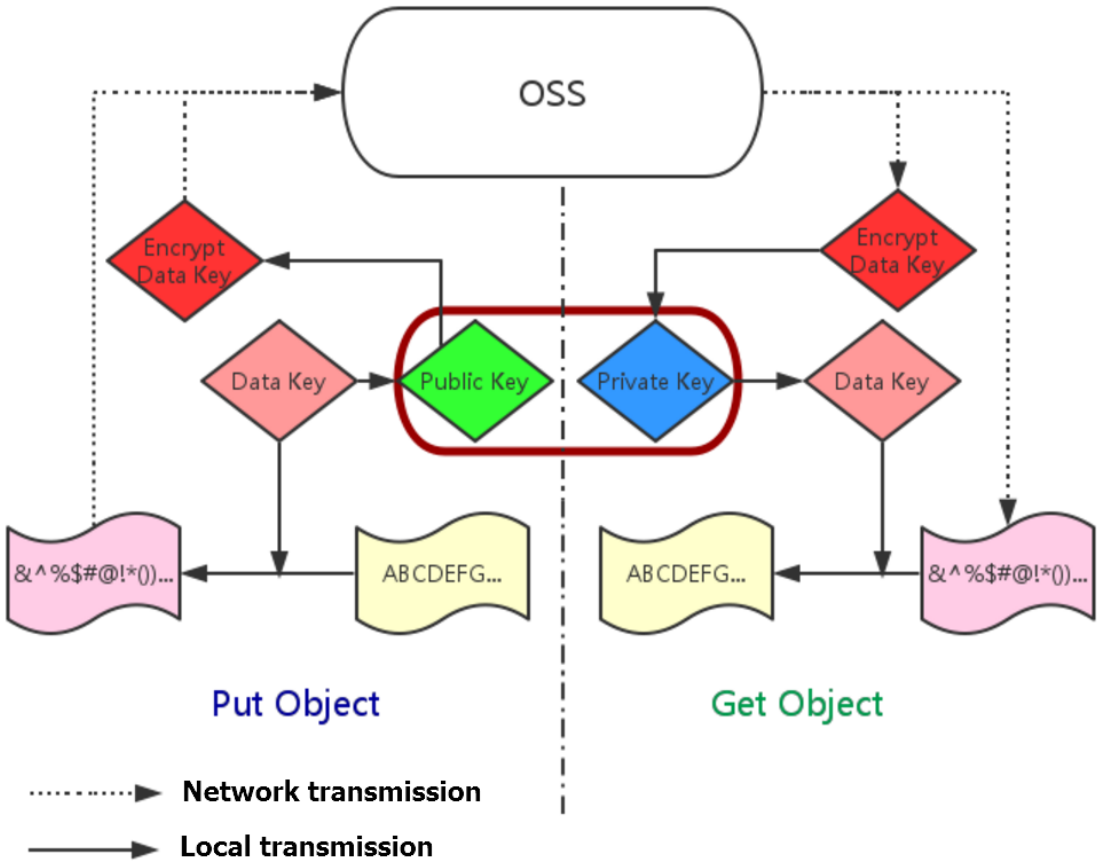
1. The user maintains a pair of RSA keys (`rsa_private_key` and `rsa_public_key`) in the local computer.

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_KEY_SIZE = 32
_AES_CTR_COUNTER_BITS_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_KEY_SIZE)
        if not self.start:
            self.start = random.randint(1, 10)
        ctr = Counter.new(_AES_CTR_COUNTER_BITS_LEN, initial_value=
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 information such as AccessKeyId, AccessKeyS
ecret, and Endpoint.
# Obtain the information through environment variables or replace the
 information such as "<Your AccessKeyId>" with the real AccessKeyId,
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 respectively.
access_key_id = os.getenv('OSS_TEST_ACCESS_KEY_ID', '<your AccessKeyId
>')
access_key_secret = os.getenv('OSS_TEST_ACCESS_KEY_SECRET', '<Your
AccessKeySecret>')
bucket_name = os.getenv('OSS_TEST_BUCKET', '<Your Bucket>')
endpoint = os.getenv('OSS_TEST_ENDPOINT', '<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_name, 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_private_key_obj = RSA.generate(2048)
rsa_public_key_obj = rsa_private_key_obj.publickey()
encrypt_obj = PKCS1_OAEP.new(rsa_public_key_obj)
decrypt_obj = PKCS1_OAEP.new(rsa_private_key_obj)
# save to local disk
file_out = open("private_key.pem", "w")
file_out.write(rsa_private_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 implemented by using the Bucket object
bucket = oss2. Bucket(oss2. Auth(access_key_id, access_key_secret),
endpoint, bucket_name)
obj_name = 'test-sig-1'
content = "test content"
#### 1 Put Object ####
# 1.1 Generate the one-time symmetric key encrypt_cipher used to
encrypt this object, where key and start are values generated at
random
encrypt_cipher = AESCipher()
# 1.2 Use the public key to encrypt the information 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_obj.encrypt
(encrypt_cipher.key))
headers['x-oss-meta-x-oss-start'] = base64.b64encode(encrypt_obj.
encrypt(str(encrypt_cipher.start)))
# 1.3. Use encrypt_cipher to encrypt the original content to obtain
encrypt_content
encryt_content = encrypt_cipher.encrypt(content)
# 1.4 Upload the object
result = bucket.put_object(obj_name, encryt_content, 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_encrypt_content = resp.read()
# 2.2 Resolve from the custom meta the key and start that are
previously used to encrypt this object
download_encrypt_key = base64.b64decode(resp.headers.get('x-oss-meta-x
-oss-key', ''))
key = decrypt_obj.decrypt(download_encrypt_key)
download_encrypt_start = base64.b64decode(resp.headers.get('x-oss-meta
-x-oss-start', ''))
start = int(decrypt_obj.decrypt(download_encrypt_start))
# 2.3 Generate the cipher used for decryption, and decrypt it to
obtain the original content
decrypt_cipher = AESCipher(key, start)
download_content = decrypt_cipher.decrypt(download_encrypt_content)
if download_content ! = content:
    print "Error!"
else:
    print "Decrypt ok. Content is: %s" % download_content
```