Alibaba Cloud Object Storage Service

Errors and Troubleshooting

Issue: 20181106

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 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 PostObject

Introduction

PostObject uploads files to OSS using forms. In Post Object, message entities are encoded in multi-form format multipart/form-data. For more information, see *RFC 2388*. In Put Object, parameters are passed by HTTP headers, while Post Object parameters are passed as form fields of the message body.

A PostObject message consists of the header and the body. The header and the body are separated by `\r\n--{boundary}`. The body consists of a series of form fields in the following format: `Content-Disposition: form-data; name="{key}"\r\n\r\n{value}\r\n--{boundary}`.

Common headers include Host, User-Agent, Content-Length, Content-Type and Content-MD5 while form fields include key, OSSAccessKeyId, Signature, Content-Disposition, object meta (x-oss-meta-*), x-oss-security-token, other HTTP headers (Cache-Control/Content-Type/Cache-Control/Content-Type/Content-Disposition/Content-Encoding/Expires/Content-Encoding/Expires) and file. The `file` must be the last field in those form fields.

For more information, see *Post Object*.

PostObject common errors

The following table shows PostObject common errors:

No.	Error	Cause	Solution
1	ErrorCode: MalformedPOSTRequest ErrorMessage: The body of your POST request is not well-formed multipart/form-data	Invalid form field format.	See PostObject form field format following the table for the correct format of form fields.
2	ErrorCode: InvalidAccessKeyId ErrorMessage: The OSS Access Key Id You provided does not exist in our records.	`AccessKeyID` was disabled or did not exist, the temporary user AccessKeyID was expired or the temporary user did not provide STS Token.	See *Invalid AccessKeyId Troubleshooting* for the troubleshooting method.

No.	Error	Cause	Solution
3	ErrorCode: AccessDenied ErrorMessage: Invalid according to Policy: Policy expired.	The `expiration` in the form field Id `policy` was expired.	Adjust `expiration` in policy while ensuring that the format of `expiration` complies with ISO8601 GMT.
4	ErrorCode: AccessDenied ErrorMessage : SignatureD oesNotMatch The request signature we calculated does not match the signature you provided. Check your key and signing method.	Incorrect signature.	See PostObject signature for the signature method.
5	ErrorCode: InvalidPol icyDocument ErrorMessage: Invalid Policy: Invalid Simple -Condition: Simple-Conditions must have exactly one property specified.	The policy contains at least one condition in the request.	See PostObject policy format.
6	ErrorCode: InvalidPol icyDocument ErrorMessage: Invalid Policy: Invalid JSON: unknown char e	Check the format of `policy` to verify if	`"` was missing and the escape character was \.
7	ErrorCode: InvalidPol icyDocument ErrorMessage: Invalid Policy: Invalid JSON: , or] expected	Incorrect `policy` format in the request.	Check if `,` or `]` was missing in policy.
8	ErrorCode: AccessDenied ErrorMessage: Invalid according to Policy: Policy Condition failed:	The `key` specified by the request and that specified by `policy` do not match.	Check the value of the form field `key` in the request.

No.	Error	Cause	Solution
	[“starts-with”, “$key”, “ user/eric/“]		
9	ErrorCode: AccessDenied ErrorMessage: Invalid according to Policy: Policy Condition failed : [“eq”, “$bucket”, “ mingdi-bjx”]	The `bucket` specified by the request and that specified by `policy` do not match.	Check the value of `bucket` in endpoint.
10	ErrorCode: AccessDenied ErrorMessage: Invalid according to Policy: Policy Condition failed : [“starts-with”, “$x-oss -meta-prop”, “prop-“]	File metadata `x-oss-meta-prop` specified by the request and that specified by policy do not match.	Check the value of `x-oss-meta-prop` in the request.
11	ErrorCode: AccessDenied ErrorMessage: Invalid according to Policy: Policy Condition failed : [“eq”, “${field}”, “${ value}”]	The `{field}` specified in form fields and that specified by policy do not match, or that field was not specified in the request.	Check the value of `{field}` in the request.
12	ErrorCode: AccessDenied ErrorMessage: You have no right to access this object because of bucket acl.	Current user did not have the required permission.	See *OSS Permission Problems and Troubleshooting*.
13	ErrorCode: InvalidArgument ErrorMessage : The bucket POST must contain the specified ‘key’. If it is specified, please check the order of the fields	The form field does not specify `key`, or it is placed after the form field `file`.	Add form field `key` or adjust orders.

- PostObject form field format

For the format of PostObject requests, note the following items:

- The header must include `Content-Type: multipart/form-data; boundary={boundary}`.
- The header and the body are separated by `\r\n--{boundary}`.
- The form field format is `Content-Disposition: form-data; name="{key}"\r\n\r\n{value}\r\n--{boundary}`.
- The form field `file` must be the last form field.
- Form field names are case-sensitive, such as policy, key, file, OSSAccessKeyId, OSSAccessKeyId, and Content-Disposition.
- When the value of `bucket` is `public-read-write`, you do not have to specify the form fields OSSAccessKeyId, policy, and Signature. If any of OSSAccessKeyId, policy, and Signature is specified, the other two form fields must be specified no matter whether `bucket` is `public-read-write` or not.

The following describes an example PostObject request:

```
POST / HTTP/1.1
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; zh-CN; rv:1.9.2
.6)
Content-Type: multipart/form-data; boundary=9431149156168
Host: mingdi-hz.oss-cn-hangzhou.aliyuncs.com
Accept: text/html, image/gif, image/jpeg, *; q=. 2, */*; q=. 2
Connection: keep-alive
Content-Length: 5052
-- 9431149156168
Content-Disposition: form-data; name="key"
test-key
--9431149156168
Content-Disposition: form-data; name="Content-Disposition"
attachment;filename=D:\img\1.png
--9431149156168
Content-Disposition: form-data; name="OSSAccessKeyId"
2NeL********j2Eb
```

Note:

- In the preceding sample request, `\r\n` shows a new line, namely a line feed. Also, this applies to the following sample requests.
- The preceding sample request is incomplete. For the complete request, see *Post Object*.

If you have any questions, see the sample code:

- *C#*
- *Java*

- PostObject policy format

  In a PostObject request, the form field `policy` is used to verify the validity of the request and it declares the conditions that must be met by the PostObject request. Specifically, those conditions are:

  - UTF-8 JSON text must be encoded with base64 before being passed into the form field `policy`.
  - The `policy` must include `expiration` and `conditions` where `conditions` must contain at least one item.

  The following shows an example `policy` before base64 encoding.

```
   {
    "expiration": "2018-01-01T12:00:00.000Z",
    "conditions": [
        ["content-length-range", 0, 104857600]
    ]
}
```

  `expiration` item specifies an expiration time of the request in the ISO8601 GMT time format. For example, `2018-01-01T12:00:00.000Z` specifies that the request must occur before 12:00 a.m. on January 1st, 2018.

  PostPolicy supports the following “conditions”:

Name	Description	Example
bucket	The bucket name of the uploaded file. Exact match is supported.	{“bucket”: “johnsmith” } or [“eq”, “$bucket”, “johnsmith”]
key	The name of the uploaded file. Exact match and prefix match are supported.	[“starts-with”, “$key”, “user/etc/“]
content-length-range	The maximum and minimum allowed sizes of the uploaded file.	[“content-length-range”, 0, 104857600]
x-oss-meta-*	The specified object meta. Exact match and prefix match are supported.	[“starts-with”, “$x-oss-meta-prop”, “prop-“]
success_action_redirect	The redirection URL upon successful upload. Exact	[“starts-with”, “$success_action_redirect”, “`http://www.aliyun.com`“]

Name	Description	Example
	match and prefix match are supported.	
success_action_status	The returned status code upon successful upload if success_action_redirect is not specified. Exact match and prefix match are supported.	[“eq”, “$success_action_status”, “204”]
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires, and so on	The HTTP headers passed as form fields. Exact match and prefix match are supported.	[“eq”, “$Content-Encoding”, “ZLIB”]

PostPolicy supports the following escape characters and uses `\` for escape.

Escape Character	Description
/	Slash
\	Backslash
“	Double quotation mark
$	Dollar sign
\b	Blank
\f	Form feed
\n	Line feed
\r	Enter
\t	Horizontal tab
\uxxxx	Unicode character

For more information about PostPolicy, see *Post Policy*.

- PostObject signature

  For a Post request to be verified, it must include AccessKeyID, policy, and Signature form fields. The signature calculation process is as follows:

  1. Create a policy encoded with `UTF-8`.
  2. Encode the policy with `base64`. The resulting value is the value to be populated into the `policy` form field, and this value is used as the string to be signed.

3. Sign the string with `AccessKeySecret`. Specifically, hash the string with hmac-sha1 and then encode it with base64. The signature method is the same as that for *Header Signature*.

Namely:

```
Signature = base64(hmac-sha1(AccessKeySecret, base64(policy)))
```

Specify the calculated signature in the form field `Signature` as follows:

```
Content-Disposition: form-data; name="Signature"
{signature}
-- 9431149156168
```

If you have any questions, see the sample code:

- *C#*
- *Java*

FAQs

- How to specify a key?

  The key is the object name, which is specified in the form field `key`. The following shows an example:

  ```
  Content-Disposition: form-data; name="key"
  {key}
  --9431149156168
  ```

- How to specify object content?

  Specify object content in the form field `file`. The following shows an example:

  ```
  Content-Disposition: form-data; name="file"; filename="images.png"
  Content-Type: image/png
  {File-content}
  -- 9431149156168
  ```

  Note:

  - The form field `file` must be the last field in a form, namely it must be placed after any other form fields.
  - `filename` is the name of the uploaded local file but not the object name.

- How to specify `content-type` of the object?

Specify `content-type` of the object in the form field `file` but not in `content-type` of the header. The following shows an example:

```
Content-Disposition: form-data; name="file"; filename="images.png"
Content-Type: image/png
{file-content}
--9431149156168
```

- How to specify `content-md5` verification for object content?

  Specify `content-md5` in the Post Object request header. Note that the MD5 value is for the entire body namely for all form fields. The following shows an example request header:

```
POST / HTTP/1.1
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; zh-CN; rv:1.9.2
.6)
Content-Type: multipart/form-data; boundary = 9431149156168
Content-MD5: tdqHe4hT/TuKb7Y4by+nJg==
Host: mingdi-hz.oss-cn-hangzhou.aliyuncs.com
Accept: text/html, image/gif, image/jpeg, *; q=. 2, */*; q=. 2
Connection: keep-alive
Content-Length: 5246
--9431149156168
```

- How to specify a signature?

  See `PostObject signature` for the signature calculation method. The signature is carried by the form field `Signature`.

- How to implement Post Object with STS Token of a temporary user?

  The usage of AccessKeyID and AccessKeySecret of a temporary user key is the same as that of a master user key and sub-user key. `Token` is carried by the form field `x-oss-security-token`. The following shows an example:

```
Content-Disposition: form-data; name="Signature"
5L0+KaeugxYygfqWLJLoy0ehOmA=
--9431149156168
Content-Disposition: form-data; name="x-oss-security-token"
{Token}
--9431149156168
```

- How to specify a callback?

  The callback is carried by the form field `callback`. The following shows an example:

```
Content-Disposition: form-data; name="callback"
eyJjYWxsYmFja0JvZHlUeXBlIjogImFwcGxpY2F0aW9uL3gtd3d3LWZvcm0t
dXJsZW5jb2RlZCIsICJjYWxsYmFja0JvZHkiOiAiZmlsZW5hbWU9JHtvYmpl
Y3R9JnNpemU9JHtzaXplfSZtaW1lVHlwZT0ke21pbWVUeXBlfSIsICJjYWxs
YmFja1VybCI6ICJodHRwOi8vb3NzLWRlbW8uYWxpeXVuY3MuY29tOjIzNDUwIn0=
--9431149156168
```

Callback custom parameters are also carried by form fields. The following shows an example:

```
Content-Disposition: form-data; name="x:var1"
{var1-value}
--9431149156168
```

- How to specify `Content-Transfer-Encoding`?

  Specify `Content-Transfer-Encoding` in the form field `file`. `file`. The following shows an example `file`form field:

  ```
  Content-Disposition: form-data; name="file"; filename="images.png"
  Content-Type: image/png
  Content-Transfer-Encoding: base64
  {file-content}
  --9431149156168
  ```

- How to specify custom meta information `Object User Meta`?

  Specify the custom meta information in form fields. The following shows an example:

  ```
  Content-Disposition: form-data; name="x-oss-meta-uuid"
  {uuid}
  --9431149156168
  Content-Disposition: form-data; name="x-oss-meta-tag"
  {tag}
  --9431149156168
  ```

  Note:

  For more information about file meta information, see *File Meta Information Object Meta*.

- How to specify conditions such as expiration, Key, Bucket, size, and header?

  PostObject for OSS supports various conditions and can meet demanding security requirements. Specify conditions in the form field `policy`. The following shows an example policy:

  ```

      "expiration": "2018-01-01T12:00:00.000Z",
      "conditions": [
          ["eq", "$bucket", "md-hz"],
          ["starts-with", "$key", "md/conf/"],
          ["content-length-range", 0, 104857600]
  ```

  In the preceding policy, the conditions for user Post Object operations are as follows:

  - `bucket` must be `md-hz`.
  - `key` must be started with `md/conf/`.

  - The size of the uploaded file must be less than 100 MB.
  - The request time must be earlier than `2018-01-01T12:00:00.000Z`.
- How to specify HTTP headers such as Cache-Control, Content-Type, Content-Disposition, Content-Encoding and Expires?

  Specify HTTP headers including `Cache-Control`, `Content-Type`, `Content-Disposition`, `Content-Encoding`, `Expires` in form fields. For the meanings of those HTTP headers, see *RFC2616* . However, `Content-MD5` needs to be specified in Post Header.

Post Object examples

- *C# Post Demo*
- *Java Post Demo*
- *JavaScript Post Demo*

Common links

- *Post object*
- *Java PostObject*

2 OSS permission

OSS errors 403

An OSS error 403 indicates that the HTTP status code returned from OSS is 403 and that the server receives your request but rejects to provide service because you have no access permission. OSS errors 403 and causes are listed in the following table:

Error	Message	Cause	Solution
SignatureDoesNotMatch	ErrorCode: SignatureDoesNotMatchErrorMessage: The request signature we calculated does not match the signature you provided. Check your key and signing method.	Client and service calculated signatures do not match	*OSS 403 errors and troubleshooting*
Postobject	ErrorCode: AccessDeniedErrorMessage: Invalid according to Policy: Policy expired.ErrorCode: AccessDenied ErrorMessage: Invalid according to Policy: Policy Condition failed: …	Invalid policy in postobject	*PostObject*
Cors	ErrorCode: AccessForbiddenErrorMessage: CORSResponse: This CORS request is not allowed. This is usually because the evalution of Origin, request method / Access-Control-Request-Method or Access-Control-Requet-Headers are	CORS is not configured or is not configured incorrectly	*OSS set up cross-domain access*

Error	Message	Cause	Solution
	not whitelisted by the resource's CORS spec .		
Refers	ErrorCode: AccessDeni edErrorMessage: You are denied by bucket referer policy.	Check the Referer configuration for the bucket	*OSS Anti-leech*
AccessDenied	See the following permissions for common errors	You have no permission.	See the following content for more information.

Among them, the permissions issue is part of the 403 error. The error with the permission problem is `AccessDenied`. These errors are described in detail below.

Common permissions errors

The privilege issue is that the current user does not have permission to specify an action. The errors returned by OSS and their causes can be found in the following table:

SN	Error	Cause
1	ErrorCode: AccessDeni edErrorMessage: The bucket you are attempting to access must be addressed using the specified endpoint. Please send all future requests to this endpoint.	The bucket does not match the endpoint.
2	ErrorCode: AccessDeni edErrorMessage: You are forbidden to list buckets.	You have no permissions for listBuckets.
3	ErrorCode: AccessDeni edErrorMessage: You do not have write acl permission on this object	You have no permissions for setObjectAcl.
4	ErrorCode: AccessDeni edErrorMessage: You do not have read acl permission on this object.	You have no permissions for getObjectAcl.

SN	Error	Cause
5	ErrorCode: AccessDeniedErrorMessage: The bucket you visit is not belong to you.	The subaccount has no permissions for bucket management like getBucketAcl, CreateBucket, deleteBucket, setBucketReferer, and getBucketReferer.
6	ErrorCode: AccessDeniedErrorMessage: You have no right to access this object because of bucket acl.	The subaccount/temporary account has no permissions to access the object like putObject getObject, appendObject, deleteObject, and postObject.
7	ErrorCode: AccessDeniedErrorMessage: Access denied by authorizer's policy.	The temporary account has no access permissions. The authorization policy specified for assuming the role of this temporary account has no permissions.
8	ErrorCode: AccessDeniedErrorMessage: You have no right to access this object.	The subaccount/temporary account has no permissions for the current operation like initiateMultipartUpload.

Permission error troubleshooting

Check whether the key is for the primary user, the subaccount or the temporary account.

- Check whether the key is for a primary user.

  Log on to the *console* to check whether the AccessKeyID exists. If it does exist, the key is for a primary user.

- Check the subaccount permission, that is, the authorization policy.

  Check the subaccount AccessKeyID and find out the corresponding subaccount by navigation to**Resource Access Management** > **User Management** > **Management** > **User Details** > **User AccessKey**.

  Log on to the console and navigate to**Resource Access Management** > **User Management** > **Management** > **User Authorization Policy** > **Individual Authorization Policy/User Authorization Policy**to check the permissions.

- Check the permissions for a temporary account.

  The AccessKeyID for the temporary account can be recognized easily since it starts with “STS”, for example, “STS.MpsSonrqGM8bGjR6CRKNMoHXe”. Log on to the console and navigate to**Resource Access Management** > **Role Management** > **Management** > **Role Authorization Policy** > **View Permissions**to check the permissions.

The access rights error process is shown in the following figure:

Procedures for checking the permissions:

1. List the required permissions and resources.
2. Check whether Action has the required operation.
3. Check whether Resource is the required operation object.
4. Check whether Effect is “Allow” instead of “Deny”.
5. Check whether Condition is set correctly.

If it is unable to detect the error through checking, the following adjustments are required:

1. The condition, if any, must be removed.
2. Remove “Deny” in Effect.
3. Change Resource to “Resource”: “*”.
4. Change Action to “Action”: “oss:*”.

Note:

- We recommend that you use the OSS authorization policy generation tool *RAM Policy Editor*to generate authorization policies.
- For more information about RAM, see *access control for Alibaba Cloud*.

3 Upload callback

About upload callback

When a file is uploaded, the OSS can provide a *Callback* to your callback server. You can carry the relevant callback parameters in the upload request to implement the upload callback. The APIs that support upload callback are *PutObject*, *PostObject*, and *CompleteMultipartUpload*. For more information, see *Developer Guide* and *API Reference*.

Note:
A callback server is also called a service server.

Application scenario

- Notification

  A typical application is to upload and callback by an authorized third party who specifies the callback parameters during file upload. After the upload is complete, the OSS sends a callback request to the callback server. When receiving the callback request, the callback server records the upload information.

- Processing, review, and statistics

  When receiving a callback request, the callback server processes, reviews, and makes statistics on the uploaded files.

Data stream

The following table describes the data streams.

Data stream	Meaning	Description
1	The client uploads a file and carries a callback parameter.[1]	The upload is implemented by SDK (*PutObject* and *CompleteMultipartUpload*, and the callback by the *PostObject* API.
2	The OSS instance stores the file and initiates a callback.	The OSS instance sends a `POST` [2] request to the specified `CallbackUrl` in the upload request. The callback time-out period is five seconds [3].

Data stream	Meaning	Description
3	The callback server returns the processing result.	• The message body returned by the callback server must be in JSON format. • The OSS determines that the callback fails if the returned result is not 200.[4]
4	The OSS returns the upload and callback result.	- • If both the upload and callback succeed, `200` is returned. • If the upload succeeds but the callback fails, `203` is returned. The value of ErrorCode is `CallbackFailed`, and ErrorMessage indicates the error cause.

Note:

- [1] For more information about the format, see `SDK/PostObject`.
- [2] For more information about the format of the callback request POST, see *Initiate a callback request*.
- [3] The time-out time is fixed and cannot be configured.
- [4] `40x` is returned for parameter invalidation or authentication failure, while `50x` is returned for time-out or connection failure.

SDK/PostObject

During the file upload, you can set the callback parameters to specify the URL of the callback server, data to be sent to the callback server, and data format. When the callback server processes a callback, some context information, such as the `bucket` and `object`, is specified using system variables. Other context information is specified using custom variables.

Upload callback parameters

The following parameters are available for an upload callback:

Field	Meaning	Description
callbackUrl	Callback server address	Required

Field	Meaning	Description
callbackHost	Value of the `Host` in the callback request message header	Optional. The default value is `callbackUrl`.
callbackBody	Callback request message body	Required. It can hold system variables and custom variables .
callbackBodyType	Value of `Content-Type` in the callback request message header, that is, the `callbackBody` data format	Optional. It can be `application/x-www-form-urlencoded` (default) or `application/json`.

Upload callback parameters are carried by the upload request in either of the following two ways:

- The callback parameters are carried by `x-oss-callback` in the message header. This is a common and recommended way.
- The callback parameters are carried by `callback` in QueryString.

Rules for generating the `x-oss-callback` or `callback` values are as follows:

```
Callback := Base64(CallbackJson)
CallbackJson := '{' CallbackUrlItem, CallbackBodyItem [, CallbackHo
stItem, CallbackBodyTypeItem] '}'
CallbackUrlItem := '"'callbackUrl'"' ':' '"'CallbackUrlValue'"'
CallbackBodyItem := '"'callbackBody'"' ':' '"'CallbackBodyValue'"'
CallbackHostItem := '"'callbackHost'"' ':' '"'CallbackHostValue'"'
CallbackBodyTypeItem := '"'callbackBodyType'"' : '"'CallbackBodyType
'"'
CallbackBodyType := application/x-www-form-urlencoded | application/
json
```

`CallbackJson` value examples are as follows:

```

    "callbackUrl" : "http://abc.com/test.php",
    "callbackHost" : "oss-cn-hangzhou.aliyuncs.com",
    "callbackBody" : "{\"bucket\":${mimeType}, \"object\":${object},\"
size\":${size},\"mimeType\":${mimeType},\"my_var\":${x:my_var}}",
    "callbackBodyType" : "application/json"
```

or

```

    "callbackUrl" : "http://abc.com/test.php",
    "callbackBody" : "bucket=${bucket}&object=${object}&etag=${etag}&
size=${size}&mimeType=${mimeType}&my_var=${x:my_var}"
```

System variables and custom variables

Variables for `CallbackJson`, such as `${bucket}`, `${object}`, and `${size}`, in the `CallbackJson` example are the OSS-defined system variables. During the callback, the OSS replaces the system variables with actual values. The following table lists the OSS-defined system variables.

Variable	Meaning
${bucket}	Storage space name
${object}	File name
${etag}	File’s etag
${size}	File size
${mimeType}	File type, such as image/jpeg
${imageInfo.height}	Image height
${imageInfo.width}	Image width
${imageInfo.format}	Image format, such as .jpg and .png

Note:

- The system variables are case sensitive.
- The system variable format must be `${bucket}`.
- imageInfo is set for images. For the non-image format, the value of imageInfo is blank.

Variables for `CallbackJson`, such as `${x:my_var}` , in the `CallbackJson` example are the custom variables. During the callback, the OSS replaces the custom variables with custom values. Custom variable values are defined and carried by the upload request in either of the following two ways:

- The custom variables are carried by `x-oss-callback-var` in the message header. This is a common and recommended way.
- The custom variables are carried by `callback-var` in QueryString.

Rules for generating the `x-oss-callback-var` or `callback-var` values are as follows:

```
CallbackVar := Base64(CallbackVarJson)
CallbackVarJson := '{' CallbackVarItem [, CallbackVarItem]* '}'
```

```
CallbackVarItem := '"''x:'VarName'"' : '"'VarValue'"'
```

`CallbackVarJson` value examples are as follows:

```

    "x:my_var1" : "value1",
    "x:my_var2" : "value2"
```

Note:

- The custom variables must start with ***x:***: They are case sensitive and in the format of `${x:my_var}`.
- The custom variable length is limited by the length of the message header and URL. We recommend that the number of the custom variables do not exceed 10 and the total length do not exceed 512 bytes.

SDK usage example

Some SDKs, such as JAVA and JS, encapsulate the preceding steps. Some SDKs, such as Python, PHP, and C, need to use the preceding rules to generate the upload callback parameters and custom variables. The following table lists SDK usage examples.

SDK	Upload callback example	Description:
JAVA	*CallbackSample.java*	Note the escape characters in `CallbackBody`.
Python	*object_callback.py*	-
PHP	*Callback.php*	`OSS_CALLBACK` and `OSS_CALLBACK_VAR` in $options do not need to be encoded using Base64, which is implemented by the SDK.
C #	*UploadCallbackSample.cs*	Use *using* to read `to read , PutObjectResult.ResponseStream` but make sure that it is disabled.
JS	*object.test.js*	-
C	*oss_callback_sample.c*	-
Ruby	*callback.rb*	-
iOS	*Callback notification after upload*	Make sure that `<var1>` the format of`var1` is x:var1.

SDK	Upload callback example	Description:
Andriod	*Callback notification after upload*	Note the escape characters in `CallbackBody`.

Note:

The Go SDK does not support upload callback currently.

PostObject usage example

PostObject supports the upload callback, whose callback parameters are carried by the form field `callback` and custom variables are carried by an independent form field. For more information, see*PostObjet*.

The following table lists PostObject usage examples.

SDK	Upload callback example
Java	*PostObjectSample.java*
Python	*object_post.py*
JS	*Javascript client signature pass-through*
C#	*PostPolicySample.cs*

Callback server

The callback server is an HTTP server that processes callback requests and POST messages sent from the OSS. The callback server URL is the value of the upload callback parameter `callbackUrl`. You can implement your own processing logic on the callback server for recording, review, processing, and statistics of the uploaded data.

Callback signature

The callback server needs to verify the signature of a POST request to make sure that the POST request is from the OSS upload callback. The callback server also can directly process the message without verifying the signature. To enhance the security of the callback server, we recommend that the callback server verify the message signature. For more information about the callback signature rules, see *Callback signature*.

Note:

The OSS callback server example describes how to implement signature verification. We recommend that you directly use the code.

Message processing

The main logic of the callback server is to process the OSS callback request. Note the following items:

- The callback server must process the POST request of the OSS.
- The OSS callback time-out time is five seconds. Therefore, the callback server must complete processing within five seconds and return the result.
- The message body sent from the callback server to the OSS must be in JSON format.
- The callback server uses its own logic, and the OSS provides examples instead of the specific service logic.

Implementation example

The following table describes the implementation examples of the callback server.

Language	Example	Running method
JAVA	*AppCallbackServer.zip*	Decompress the package and run `java -jar oss-callback-server-demo.jar 9000`.
PHP	*callback-php-demo.zip*	Deploy and run the program to in **Apache** environment.
Python	*callback_app_server.py.zip*	Decompress the package and run `python callback_app_server.py`.
Ruby	*oss-callback-server*	Run `ruby aliyun_oss_callback_server.rb`.

Debugging procedure

The upload callback debugging includes debugging of the client that uploads a file and the callback server that processes the callback. We recommend that you debug the client first and then the callback server. After independently debugging the two parts, perform the complete upload callback.

- Client debugging

  You can use the callback server `http://oss-demo.aliyuncs.com:23450` provided by the OSS, that is, the callback parameter `callbackUrl` to debug the client. The callback server only verifies the callback request signature, and does not process the callback request. For callback requests whose signatures are successfully verified, the callback server returns `{"Status":"OK"}`. For callback requests whose signatures fail to be verified, the callback server returns `400 Bad Request`. For non-POST requests, the callback server returns `501 Unsupported method`. For more information about the code of the callback server example, see *callback_app_server.py.zip*.

- Callback server debugging

The callback server is an HTTP server that can process the POST request. You can modify the callback server based on the example provided by the OSS or implement it by yourself. The following table describes the examples of the callback server provided by the OSS.

Language	Example	Running method
JAVA	*AppCallbackServer.zip*	Decompress the package and run `java -jar oss-callback-server-demo.jar 9000`.
PHP	*callback-php-demo.zip*	Deploy and run the program to in **Apache** environment
Python	*callback_app_server.py.zip*	Decompress the package and run `python callback_app_server.py`.
C#	*callback-server-dotnet.zip*	Compile the program and run `aliyun-oss-net-callback-server.exe 127.0.0.1 80`.
Go	*callback-server-go.zip*	Compile the program and run `aliyun_oss_callback_server`.
Ruby	*oss-callback-server*	Run `ruby aliyun_oss_callback_server.rb`.

The callback server can be debugged by running the **cURL** command. The following commands may be used:

```
# Run the following command to send a `POST` request whose message
body is `object=test_obj` to the callback server:
curl -d "object=test_obj" http://oss-demo.aliyuncs.com:23450 -v
# Run the following command to send a `POST` request whose message
body is `post.txt` to the callback server:
curl -d @post.txt http://oss-demo.aliyuncs.com:23450 -v
```

```
# Run the following command to send a `POST` request whose message
body is `post.txt` and which carries the specified message header `
Content-Type` to the callback server:
curl -d @post.txt -H "Content-Type: application/json" http://oss-
demo.aliyuncs.com:23450 -v
```

Note:

- When debugging the callback server, ignore signature verification because it is difficult for `cURL` to simulate the signature function.
- The OSS example already provides the signature verification function. We recommend that you directly use it.
- We recommend that the callback server provide the logging function to record all messages, facilitating debugging and tracking.
- After correctly processing a callback request, the callback server must return `200` instead of 20x.
- The message body sent from the callback server to the OSS must be in JSON format, and `Content-Type` is set to`application/json`.

Common errors and causes

- InvalidArgument

```
<Error>
  <Code>InvalidArgument</Code>
  <Message>The callback configuration is not json format.</Message>
  <RequestId>587C79A3DD373E2676F73ECE</RequestId>
  <HostId>bucket.oss-cn-hangzhou.aliyuncs.com</HostId>
  <ArgumentName>callback</ArgumentName>
  <ArgumentValue>{"callbackUrl":"8.8.8.8:9090","callbackBody
":"{"bucket":${bucket},"object":${object}}","callbackBodyType":"
application/json"}</ArgumentValue>
</Error>
```

Cause:

The callback parameter settings are incorrect, or the parameter format is incorrect. The common error is that the callback parameters in `ArgumentValue` are not in valid JSON format. In JSON, `\` and `"` are escape characters. For example, `"callbackBody":"{"bucket":${bucket},"object":${object}}"` must be `"callbackBody":"{\"bucket\":${bucket},\"object\":${object}}"`. For more information about the SDKs, see the upload callback examples in the *SDK usage example* part.

Character after escape	Character before escape
\\	\\\\
“	\\\”
\b	\\b
\f	\\f
\n	\\n
\r	\\r
\t	\\t

- CallbackFailed

```
<Error>
  <Code>CallbackFailed</Code>
  <Message>Response body is not valid json format.</Message>
  <RequestId>587C81A125F797621829923D</RequestId>
  <HostId>bucket.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
```

Cause:

The message body sent from the callback server to the OSS is not in JSON format. You can confirm the content by running `curl -d "<Content>" <CallbackServerURL> -v` or capture packets. We recommend that you use Wireshark to capture packets in Windows, and use **`tcpdump`** to capture packets in Linux. Invalid returned messages include: `OK` and `\357\273\277{"Status":"OK"}` (the BOM header containing the `ef bb bf` bytes).

```
<Error>
  <Code>CallbackFailed</Code>
  <Message>Error status : -1. OSS can not connect to your callbackUr
l, please check it.</Message>
  <RequestId>587C8735355BE8694A8E9100</RequestId>
  <HostId>bucket.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
```

Cause:

The processing time of the callback server exceeds five seconds. Therefore, the OSS determines that a time-out occurs. We recommend that you modify the processing logic of the callback server to asynchronous processing to make sure that it can complete processing within five seconds and returns the result to the OSS.

```
<Error>
  <Code>CallbackFailed</Code>
  <Message> error status:-1 8.8.8.8: 9090 reply timeout, cost: 5000
MS, timeout: 5000 MS (Ernest-4, errno170) </message>
```

```
  <RequestId>587C8D382AE0B92FA3EEF62C</RequestId>
  <HostId>bucket.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
```

Cause:

The processing time of the callback server exceeds five seconds. Therefore, the OSS determines that a time-out occurs.

```
<Error>
  <Code>CallbackFailed</Code>
  <Message>Error status : 400.</Message>
  <RequestId>587C89A02AE0B92FA3C7981D</RequestId>
  <HostId>bucket.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
```

Cause:

The status code of the message sent from the callback server to the OSS is `400`. Check the processing logic of the callback server.

```
<Error>
  <Code>CallbackFailed</Code>
  <Message>Error status : 502.</Message>
  <RequestId>587C8D382AE0B92FA3EEF62C</RequestId>
  <HostId>bucket.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
```

Cause:

The callback server is not started, `CallbackUrl` is missing in the callback parameters, or the network between the OSS instance and the callback server is disconnected. We recommend that you deploy the callback server on the ECS, which belongs to the same intranet as the OSS, to save the traffic cost and guarantee the network quality.

Common links

- *Callback Guide*
- *Callback API*

4 OSS 403

Error: UserDisable.UserDisable

If the following error UserDisable is reported when you access OSS:

```
<Code>UserDisable</Code>
                <Message> userdisable </message>
```

The error may caused by two possible reasons:

- Access is denied due to account outstanding payment.

  You can click **Billing Management** on the *OSS console* to check whether an outstanding payment is made. If any, recharge the account in time.

  You can click **Billing Management** on the *OSS console* to check whether an outstanding payment is made. If any, recharge the account in time.

  Note:

  - Even if an outstanding payment is made, you still can use OSS for 24 hours and your access will be banned later.
  - Your historical data is kept for 15 days and will be deleted later.
  - Once you see an “Alibaba Cloud OSS Arrearage Message” in the Message Center, recharge your account in time. If not, you cannot use OSS.
- Access is denied due to security reasons.

  Click **Notice** on the console to enter the **Message Center** and check the notice of violation on the Security message on the right side. Violation may be caused by various of reasons.

  Note:
  If your account is banned, you must do whatever necessary to recover the use of your account. A new account does not guarantee your normal use.

Error: RequestTimeTooSkewed.The difference between…

If the following error RequestTimeTooSkewed is reported when you access OSS:

```
<Code>RequestTimeTooSkewed</Code>
```

```
<Message>The difference between the request time and the current time
is too large.</Message>
```

The cause is that the interval between your request time and the time at which OSS receives your request exceeds 15 minutes. Therefore, OSS considers this request to be invalid due to security reasons and returns this error. You must check the system time of the device sending the request , and adjust it to a correct time according to the time zone.

You may have the following questions:

- What are the criteria for adjusting the system time of the machine or device sending the request?

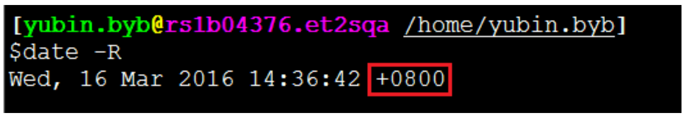
  The system time adopted by OSS is the Greenwich Mean Time (GMT). Therefore, the system time of your device must be adjusted to GMT or to a time within a time zone corresponding to GMT. GMT is the zone time of zero zone, that is the World Standard Time.

  If, for example, the system of your device that accesses OSS is configured with GMT+08:00 , the system time must be adjusted to a time that is 8 hours earlier than GMT. The other time can be adjusted similarly. The standard time in China is Beijing Time, that is GMT+08:00. If your system time is located at GMT+08:00, your system time only needs to be adjusted to Beijing Time.

  - To check your time zone using the Windows system,

    click**Control Panel** > **Clock, Language, and Region** > **Set Date and Time** to open the date and time. The +08:00 in the Time Zone column indicates that your device is located in the time zone GMT+08:00.
  - If your system is Linux/Unix,

    run the `date -R` command to check the time and the time zone. +0800 is shown in the following figure, which indicates that the system time zone of your device is GMT+08:00.

```
[yubin.byb@rs1b04376.et2sqa /home/yubin.byb]
$date -R
Wed, 16 Mar 2016 14:36:42 +0800
```

- Is there a problem of time synchronization when using OSS across multiple regions like Hangzhou, Singapore, and the United States?

  There is certainly no problem. The OSS in each region uses GMT and the system time of your device sending the request is also GMT.

Error: InvalidAccessKeyId.The OSS Access Key Id...

If the following error is reported when you access OSS:

```
<Code>InvalidAccessKeyId</Code>
<Message>The OSS Access Key Id you provided does not exist in our
records.</Message>
```

The possible cause is that your AccessKeyID is disabled or does not exist. You can troubleshoot the error as follows:

Log on to *AccessKey management* on the Alibaba Cloud console to confirm that the AccessKeyID used for accessing OSS does exist and has been activated.

- If your AccessKeyID is disabled, activate it.
- If your AccessKeyID does not exist, create a new AccessKeyID and use it to access OSS.

Error: AccessDenied.The bucket you are attempting to...

If the following error is reported when you access OSS:

```
<Code>AccessDenied</Code>
<Message>The bucket you are attempting to access must be addressed
using the specified endpoint. Please send all future requests to this
endpoint.</Message>
```

The cause is that the endpoint you use to access the bucket is incorrect. For endpoint details, see *OSS basic concepts*.

How can we find out a correct endpoint? If the SDK is abnormal as follows or returns the following error:

```
<Error>
  <Code>AccessDenied</Code>
  <Message>The bucket you are attempting to access must be addressed
using the specified endpoint. Please send all future requests to this
endpoint.</Message>
  <RequestId>56EA****3EE6</RequestId>
  <HostId>my-oss-bucket-*****.aliyuncs.com</HostId>
  <Bucket>my-oss-bucket-***</Bucket>
  <Endpoint>oss-cn-****.aliyuncs.com</Endpoint>
</Error>
```

- Then `oss-cn-****.aliyuncs.com` in the `endpoint` is the correct endpoint. You must use `http://oss-cn-****.aliyuncs.com` or `https`：//`oss-cn-****.aliyuncs.com` as the endpoint to access OSS.
- If the `endpoint` is not shown in the error returned, you must log on to OSS console, and on the **Overview** page find out the bucket you are attempting to access. Then click the bucket to

enter the **Bucket Overview**page. On the **`OSS Domain Name`** area, you can see the domain names of the intranet and the Internet.

- The Internet domain name is used to access OSS on the Internet. The intranet domain name is used to internally access OSS on the intranet of Alibaba Cloud. For example, if you access OSS on your ECS, you can use an intranet domain name.
- Endpoint is composed of the domain name (excluding the bucket part) and the access protocol. For example, the Internet domain name of OSS in the preceding picture is `oss-****.aliyuncs.com` Therefore, the Internet endpoint is `http://oss-cn-****.aliyuncs.com` and similarly its intranet endpoint is `http://oss-cn-****-internal.aliyuncs.com`.

Error: ImageDamage.The image file may be damaged

If the following error is reported when you access OSS:

```
<Code>ImageDamage</Code>
<Message>The image file may be damaged.</Message>
```

This error indicates that part of the image file message is lost or damaged, and the image cannot be identified or processed. You may have a question that an image can be processed locally by an image processor but the OSS reports an error. The cause is that the image processor does some processing of the damaged image but the OSS service currently does not have this function.

Error: AccessDenied.AccessDenied

If the following error is reported when you access OSS:

```
<Code>AccessDenied</Code>
<Message>AccessDenied</Message>
```

This error indicates that the user accessing OSS has no permissions for the current operation. The correct `AccessKeyID/AccessKeySecret` must be used. If the account you are using is a subaccount/temporary account (STS), you must confirm your current permissions.

Confirmation method:

Check your permissions on the *RAM console*. Click**User management** and click **User who needs to confirm the permission**, then click **User Authorization Policy** and **Authorization Policy for Group**. Confirm the current account has been granted the permissions to operate on the bucket/object.

Error: SignatureDoesNotMatch. The request signature we calculated…

If the following error is reported when you access OSS:

```
<Code>SignatureDoesNotMatch</Code>
<Message>The request signature we calculated does not match the
signature you provided. Check your key and signing method.</Message>
```

Troubleshoot the error as follows:

1. Check the endpoint.

   Check whether there is a bucket before the endpoint, whether there is unnecessary `/` behind the endpoint, and whether there are unnecessary `spaces` at two sides of the endpoint. For example, the endpoint `http://my-bucket.oss-cn-hangzhou.aliyuncs.com` and `http://oss-cn-hangzhou.aliyuncs.com/` are invalid, while `http:// oss-cn-hangzhou.aliyuncs.com` and `https:// oss-cn-hangzhou.aliyuncs.com` are valid domain names.
2. Check the AccessKeyID/AccessKeySecret.

   Confirm that the AccessKeyID/AccessKeySecret is correct. Make sure there are no spaces at two sides of the AccessKeyID/AccessKeySecret, especially when it is copied and pasted.
3. Check the BucketName/ObjectKey.

   Make sure that the BucketName/ObjectKey is valid and compliant with the naming rule.

   - Bucket nomenclature: The name of a bucket only consists of lower-case letters, numbers , and hyphens (-) and must start with a lower-case letter or number.The length must be between 3 bytes and 63 bytes.
   - Object nomenclature: The name of an object adopts UTF-8 codes with a length of 1 to 1,023 bytes. The name cannot start with “/“ or “\”.
4. If your own signature is used, you must follow the signature method provided by OSS SDK.

   OSS SDK supports URL/Header signatures. For more information, see the SDK documentation .
5. If your environment is not suitable for SDK use but you do need to use your signature, see *User signature verification* for the signature method. You must check each signature field carefully.

   A visual signature tool is provided on the OSS forum. You must compare each signature field and the final signature. The signature tool is available at the *Signature tool address*.

6. If you use a proxy, you must check whether the proxy server has been configured with an additional header.

Other errors

You must judge the causes based on the error codes and messages returned from the SDK. The error messages indicate the error causes. If you suspect that the error is related with the network environment, you can use *ossprobe* for error troubleshooting and the Ossprobe may give possible causes.