ALIBABA CLOUD

阿里云

对象存储 OSS
常见错误排查

文档版本：20220328

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.网络超时处理 05
2.Post Object错误及排查 08
3.OSS权限问题及排查 17
4.OSS防盗链（Referer）配置及错误排除 20
5.上传回调错误及排除 23
6.STS常见问题及排查 33
7.OSS跨域资源共享（CORS）错误及排除 35
8.OSS 403错误及排查 37
9.自签名计算失败 41
10.视频播放异常 44

1.网络超时处理

本文介绍如何处理在使用OSS SDK时可能出现的网络超时问题。

问题现象

OSS SDK在使用过程中出现的较典型的问题是使用SDK联网过程中出现超时，且在上传过程中提示ConnectionTimeOut的错误，比较影响用户使用体验。

问题排查

由于该问题无法复现，现列举以下可能的原因进行逐一排查，以解决OSS SDK的网络超时问题。

1. 网络环境

   分析网络路径：

   手机 / PC --- 运营商网络 --- OSS Server

   用户所在的网络环境可能处在运营商网络边缘节点，向运营商网络请求成功率比较低。可以利用CDN的边缘加速节点，减少手机 / PC网络对运营商网络的依赖。具体链路如下：

   手机 / PC -- CDN就近节点 -- 运营商网络 -- OSS Server

   具体的技术方案参见：OSS如何开启CDN加速服务

   如果问题没有完全解决，仍有ConnectionTimeOut的错误提示，请查看网络配置的分析。

2. 网络配置

   以下代码为具体的超时错误信息：

   ```
   "ConnectionTimeoutError&errormsg=Failed to upload some parts with error: ConnectionTimeoutError: Connect timeout for 60000ms, PUT https://***.oss-cn-hangzhou.aliyuncs.com/***/***/***.mp4?partNumber=2&uploadId=*** -2 (connected: false, keepalive socket: false)headers: {} part_num: 2
   ```

   从错误信息中可以得出如下结论：

   - 超过60s没有收到服务器的返回信息而断开连接。
   - 分析CDN提供的日志，超时的主要原因是分片还没上传成功，连接就已经断开。
   - 如果上传的文件较大，在弱网络环境下，客户端/PC端发送的请求迟迟得不到OSS服务器的响应。

   综合以上分析过程，现推荐以下几种解决方案：

   - 上传方式采用分片断点上传，每个分片的大小不超过1MB。
   - 添加重传机制，保证某一个分片上传失败后还可以继续上传。
   - 增大超时时间。

```
//js sdk 分片断点上传示例代码
let retryCount = 0;
let retryCountMax = 3;
...
const uploadFile = function uploadFile(client) {
  if (!uploadFileClient || Object.keys(uploadFileClient).length === 0) {
    uploadFileClient = client;
  }
  ...
  console.log(`${file.name} => ${key}`);
  const options = {
    progress,
    partSize: 1000 * 1024,//设置分片大小
    timeout: 120000,//设置超时时间
  };
  if (currentCheckpoint) {
    options.checkpoint = currentCheckpoint;
  }
  return uploadFileClient.multipartUpload(key, file, options).then((res) => {
    console.log('upload success: %j', res);
    currentCheckpoint = null;
    uploadFileClient = null;
  }).catch((err) => {
    if (uploadFileClient && uploadFileClient.isCancel()) {
      console.log('stop-upload!');
    } else {
      console.error(err);
      //retry
      if (retryCount < retryCountMax){
          retryCount++;
          console.error("retryCount : " + retryCount);
          uploadFile('');
      }
    }
  });
};
```

总结

如果通过标准OSS访问域名（例如oss-cn-hangzhou.aliyuncs.com）访问OSS的数据，且这是通过运营商网络对OSS进行访问，由于网络环境的复杂性，例如在一些弱网络环境下或网络不稳定的情况，就会在上传过程中遇到ConnectionTimeOut的网络错误。可以尝试通过以下方式来解决：

- 采用分片断点上传，每个分片的大小不要超过1MB，也不要小于100KB。

  > **说明** OSS服务器不接受小于100KB大小的分片。

- 添加重传机制，保证某一个分片上传失败后还可以继续上传。

  > **说明** Android/iOS SDK中已经默认开启，不需要额外配置。

- 增大超时时间。

- 通过CDN全站加速服务来提升OSS传输。

  详情请参见OSS如何开启CDN加速服务。

2.Post Object错误及排查

本文介绍Post Object时的常见错误及解决方法。

Post Object简介

Post Object使用表单上传文件到OSS。Post Object的消息实体通过多重表单格式multipart/form-data编码，详细说明请参看RFC 2388。PutObject中参数通过HTTP请求头传递，Post Object参数则作为消息体的表单域传递。

Post Object消息包括消息头（Header）和消息体（Body）。Header和Body之间，由 `\r\n--{boundary}` 分割。Body由一系列的表单域构成，表单域格式如下：

```
Content-Disposition: form-data; name="{key}"\r\n\r\n{value}\r\n--{boundary}
```

常见的Header有Host、User-Agent、Content-Length、Content-Type、Content-MD5等，表单域字段有key、OSSAccessKeyId、Signature、Content-Disposition、object meta(x-oss-meta-*)、x-oss-security-token、其它HTTP Header(Cache-Control/Content-Type/Cache-Control/Content-Type/Content-Disposition/Content-Encoding/Expires/Content-Encoding/Expires)、file等。表单域中 `file` 必须是最后一个。

更多详细的介绍请参看Post Object。

Post Object常见错误

Post Object常见错误见下表：

序号	错误	原因	解法
1	ErrorCode: MalformedPOSTRequest ErrorMessage: The body of your POST request is not well-formed multipart/form-data	表单域格式不符合要求	表单域格式请参看表后的Post Object表单域格式
2	ErrorCode: InvalidAccessKeyId ErrorMessage: The OSS Access Key Id you provided does not exist in our records.	`AccessKeyID` 禁用或不存在，或者过期的临时用户AccessKeyID，或者临时用户没有提供STS Token	排查方法请参看InvalidAccessKeyId错误排查
3	ErrorCode: AccessDenied ErrorMessage: Invalid according to Policy: Policy expired.	表单域policy中的 `expiration` 过期	请调整policy中的 `expiration` ，注意expiration的格式ISO8601 GMT

序号	错误	原因	解法
4	ErrorCode: AccessDenied ErrorMessage: SignatureDoesNotMatch The request signature we calculated does not match the signature you provided. Check your key and signing method.	签名错误	签名方法请参看下面的Post Object的签名
5	ErrorCode: InvalidPolicyDocument ErrorMessage: Invalid Policy: Invalid Simple-Condition: Simple-Conditions must have exactly one property specified.	请求中policy至少包含一个condition	请参看表后的Post Object的Policy格式
6	ErrorCode: InvalidPolicyDocument ErrorMessage: Invalid Policy: Invalid JSON: unknown char e	请求中的 `policy` 格式不正确	请检查policy格式， `"` 是否加缺失，转义字符是否加 `\`
7	ErrorCode: InvalidPolicyDocument ErrorMessage: Invalid Policy: Invalid JSON: , or] expected	请求中的 `policy` 格式不正确	请检查policy中是否缺少 `,` 或 `]`
8	ErrorCode: AccessDenied ErrorMessage: Invalid according to Policy: Policy Condition failed: [“starts-with”, “$key”, “user/eric/ “]	请求指定的 `key` 与 `policy` 限定的不符	请检查请求中表单域 `key` 的值
9	ErrorCode: AccessDenied ErrorMessage: Invalid according to Policy: Policy Condition failed: [“eq”, “$bucket”, “mingdi-bjx”]	请求指定 `bucket` 与 `policy` 限定的不符	请检查endpoint中的 `bucket` 的值

序号	错误	原因	解法
10	ErrorCode: AccessDenied ErrorMessage: Invalid according to Policy: Policy Condition failed: [“starts-with”, “$x-oss-meta-prop”, “prop- “]	请求指定的文件元数据 `x-oss-meta-prop` 与policy限定的不符	请检查请求中的 `x-oss-meta-prop` 的值
11	ErrorCode: AccessDenied ErrorMessage: Invalid according to Policy: Policy Condition failed: [“eq”, “${field}”, “${value}”]	表单域中指定的 `{field}` 与policy中限定的值不符，或者在请求中没有指定	请检查请求中的 `{field}` 的值
12	ErrorCode: AccessDenied ErrorMessage: You have no right to access this object because of bucket acl.	当前用户无权限	请参看OSS权限问题及排查
13	ErrorCode: InvalidArgument ErrorMessage: The bucket POST must contain the specified ‘key’. If it is specified, please check the order of the fields	表单域没有指定 `key`，或者放在了表单域 `file` 后	请添加表单域 `key` 或调整顺序

- Post Object表单域格式

  Post Object请求格式，有以下注意点：

  - Header一定要有 `Content-Type: multipart/form-data; boundary={boundary}` 。
  - Header和body之间由 `\r\n--{boundary}` 分割。
  - 表单域格式：

    ```
    Content-Disposition: form-data;
            name="{key}"\r\n\r\n{value}\r\n--{boundary}
    ```

  - 表单域名称大小写敏感，如policy、key、file、OSSAccessKeyId、OSSAccessKeyId、Content-Disposition。

    > 注意 表单域 `file` 必须为最后一个表单域。

  - Bucket为 `public-read-write` 时，可以不指定表单域OSSAccessKeyId、policy、Signature，一旦指定OSSAccessKeyId、policy、 Signature中的任意一个，无论bucket是否为public-read-write，则另两个必须指定。

下面是Post Object请求的示例：

```
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
```

> **说明**
> - 上面示例请求中 `\r\n` 显示为新行，即换行，后面的示例请求类似。
> - 上面的示例为请求的部分内容，完整的请求请参看Post Object。

如果您还有疑问，请参考示例代码：

- C#
- Java
- JS

- Post Object的Policy格式

  Post Object请求的 `policy` 表单域用于验证请求的合法性，声明了Post Object请求必须满足的条件。限定条件为：

  - UTF-8格式的JSON文本，经过base64编码后放在表单域policy中。
  - Policy中必须包含expiration和condtions，其中condtions至少有一项。

  下面是base64编码前的policy示例：

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

  `expiration` 项指定了请求的过期时间， ISO8601 GMT 时间格式；如 `2018-01-01T12:00:00.000Z` 指定请求必须发生在2018年1月1日12点前。

  Post Policy支持的限定条件（Conditions）如下：

名称	描述	示例
bucket	上传文件的Bucket名称。支持精确匹配方式。	{ “bucket” : “johnsmith” } 或 [“eq” , “$bucket” , “johnsmith”]
key	上传文件的名称。支持精确匹配和前缀匹配方式。	[“starts-with” , “$key” , “user/etc/]”
content-length-range	上传文件允许的最小、最大长度。	[“content-length-range” , 0, 104857600]
x-oss-meta-*	指定的object meta。支持精确匹配和前缀匹配方式。	[“starts-with” , “$x-oss-meta-prop” , “prop- “]
success_action_redirect	上传成功后的跳转URL地址。支持精确匹配和前缀匹配方式。	[“starts-with” , “$success_action_redirect” , “http://www.aliyun.com”]
success_action_status	未指定success_action_redirect时，上传成功后的返回状态码。支持精确匹配和前缀匹配方式。	[“eq” , “$success_action_status” , “204”]
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires等	HTTP请求头，作为表单域传递。支持精确匹配和前缀匹配方式。	[“eq” , “$Content-Encoding” , “ZLIB”]

Post Policy有以下转义字符，使用 `\` 转义。

转义字符	描述
/	斜杠
\	反斜杠
“	双引号
$	美元符
\b	空格
\f	换页
\n	换行
\r	回车
\t	水平制表符
\uxxxx	Unicode字符

Post Policy更详细的说明，请参考 Post Policy。

- Post Object的签名

对于验证的Post请求，请求中必须包含AccessKeyID、policy、Signature表单域。计算签名的流程如下：

i. 创建一个 `UTF-8` 编码的policy。
ii. 将policy进行 `base64` 编码，其值即为policy表单域该填入的值，并将该值作为将要签名的字符串。
iii. 使用 `AccessKeySecret` 对要签名的字符串进行签名，先用hmac-sha1哈希，再base64编码；签名方法与Header签名的方法相同。

即：

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

计算出的签名在表单域 `Signature` 中指定，如下所示：

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

如果您还有疑问，请参考示例代码：

- C#
- Java
- JS

常见问题

- 怎么指定key?

  key即object name，在表单域 `key` 中指定，示例如下：

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

- 怎么指定object内容?

  Object内容通过表单域 `file` 中指定，示例如下：

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

  > **说明**
  > - 表单域 `file` 必须是表单中的最后一个域，即表单域 `file` 必须放在所有表单域后。
  > - `filename` 是上传的本地文件名称，而不是object name。

- 怎么指定object类型content-type?

  Object类型在表单域 `file` 中指定 `Content-Type` ，而不是Header中的 `Content-Type` ，示例如下：

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

- 怎么指定object内容md5校验content-md5?

  在Post Object请求头中指定 `Content-MD5` ，注意MD5值是整个body的MD5，即所有表单域的MD5。请求Header示例如下：

```
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
```

- 怎么指定签名Signature?

  签名的计算方法请参看 `PostObject中的签名` ， 签名通过表单域 `Signature` 携带。

- 怎么使用临时用户STS Token执行Post Object?

  临时用户密钥的AccessKeyID、AccessKeySecret用法跟主用户、子用户相同，Token放在表单域 `x-oss-security-token` 中携带。示例如下：

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

> **说明** 如果您想了解更多访问控制的信息，请参看阿里云访问控制。

- 怎么指定上传回调（callback）?

  上传回调(callback)通过表单域 `callback` 携带。示例如下：

```
Content-Disposition: form-data; name="callback"
eyJjYWxsYmFja0JvZHlUeXBlIjogImFwcGxpY2F0aW9uL3gtd3d3LWZvcm0tdXJsZW5jb2RlZCIsICJjYWxsYmFja0JvZHkiOiAiZmlsZW5hbWU9JHtvYmplY3R9JnNpemU9JHtzaXplfSZtaW1lVHlwZT0ke21pbWVUeXBlfSIsICJjYWxsYmFja1VybCI6ICJodHRwOi8vb3NzLWRlbW8uYWxpeXVuY3MuY29tOjIzNDUwIn0=
--9431149156168
```

  Callback的自定义参数也是通过表单域携带。示例如下：

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

> **说明** 如果您想了解callback更多内容，请参看上传回调。

- 怎么指定Content-Transfer-Encoding?

  在表单域 `file` 中指定 `Content-Transfer-Encoding` 。 `file` 表单域示例如下：

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

- 怎么指定用户自定义元信息Object User Meta?

  用户自定义元信息，可以通过表单域指定，示例如下：

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

> **说明** 文件元信息更详细的说明，请参看文件元信息Object Meta。

- 怎么指定限定条件expiration、Key、Bucket、size、header等?

  OSS的Post Object支持丰富的条件限制，可以满足高安全性要求。限定条件Conditions可以通过表单域 `policy` 指定，详细的说明请参看上面的Post Object的Policy格式。下面是一个policy的示例：

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

  上面的policy实例中，用户的Post Object操作的限定条件如下：

  - `bucket` 必须是 `md-hz` 。
  - `key` 必须以 `md/conf/` 开头。
  - 上传的文件长度必须在100M以下。
  - 请求时间在 `2018-01-01T12:00:00.000Z` 之前。

- 怎么指定Cache-Control、Content-Type、Content-Disposition、Content-Encoding、Expires等HTTP Header?

  `Cache-Control` `Content-Type` 、 `Content-Disposition` 、 `Content-Encoding` `Expires` 等HTTP Header需要在表单域中指定，这些HTTP Header的含义请参看RFC2616 。但是 `Content-MD5` 需要在Post Header中指定。

Post Object示例

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

常用链接

- Post Object
- Java PostObject实现

3.OSS权限问题及排查

本文介绍OSS权限问题的排查方法及解决方案。

OSS 403问题

OSS 403指OSS返回的HTTP状态码是403，可以简单的理解为没有权限访问，服务器收到请求但拒绝提供服务。OSS 403错误及原因如下表：

错误	错误码错误信息	错误原因	解决办法
SignatureDoesNotMatch	ErrorCode: SignatureDoesNotMatch ErrorMessage: The request signature we calculated does not match the signature you provided. Check your key and signing method.	客户端和服务计算的签名不符	OSS 403错误及排查
PostObject	ErrorCode: AccessDeniedErrorMessage: Invalid according to Policy: Policy expired.ErrorCode: AccessDenied ErrorMessage: Invalid according to Policy: Policy Condition failed: ...	PostObject中Policy无效	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 not whitelisted by the resource's CORS spec.	CORS没有配置或配置不对	OSS设置跨域访问
Refers	ErrorCode: AccessDeniedErrorMessage: You are denied by bucket referer policy.	请检查Bucket的Referer配置	OSS防盗链
AccessDenied	见以下权限常见错误	无权限	下面详细讲述

其中，权限问题是403错误的一部分。权限问题的错误是 `AccessDenied` 。以下将详细讲述这类错误。

权限常见错误

权限问题是当前用户没有指定操作的权限。OSS返回的错误及原因见下表：

序号	错误	原因
1	ErrorCode: AccessDeniedErrorMessage: The bucket you are attempting to access must be addressed using the specified endpoint. Please send all future requests to this endpoint.	Bucket和Endpoint不符
2	ErrorCode: AccessDeniedErrorMessage: You are forbidden to list buckets.	无listBuckets权限
3	ErrorCode: AccessDeniedErrorMessage: You do not have write acl permission on this object	无setObjectAcl权限
4	ErrorCode: AccessDeniedErrorMessage: You do not have read acl permission on this object.	无getObjectAcl权限
5	ErrorCode: AccessDeniedErrorMessage: The bucket you access does not belong to you.	子用户没有Bucket管理的权限（如getBucketAcl CreateBucket、deleteBucket setBucketReferer、getBucketReferer等）
6	ErrorCode: AccessDeniedErrorMessage: You have no right to access this object because of bucket acl.	子用户/临时用户没有访问Object的权限（如putObject getObject、appendObject deleteObject、postObject）等
7	ErrorCode: AccessDeniedErrorMessage: Access denied by authorizer's policy.	临时用户访问无权限，该临时用户角色扮演指定授权策略，该授权策略无权限
8	ErrorCode: AccessDeniedErrorMessage: You have no right to access this object.	子用户/临时用户无当前操作权限（如initiateMultipartUpload等）

权限问题排查

辨别密钥是主用户、子用户还是临时用户的。

- 是否是主用户的密钥：

需要到控制台查看AccessKeyID是否存在，如果存在说明是主用户。

- 查看子用户的权限，即该子用户的授权策略：

  在控制台**访问控制 > 用户管理 > 管理 > 用户详情 > 用户AccessKey**，查看子用户的AccessKeyID，并找到对应的子用户；在控制台**访问控制 > 用户管理 > 管理 > 用户授权策略 > 个人授权策略/加入组的授权策略 > 查看子账户的权限**。

- 查看临时用户的权限，即对应角色的权限：

  临时用户的密钥的AccessKeyID，以STS开头比较好辨认，如“STS.MpsSonrqGM8bGjR6CRKNMoHXe”。在控制台**访问控制 > 角色管理 > 管理 > 角色授权策略 > 查看权限**。

权限检查流程如下：

1. 列出需要的权限和资源。
2. 检查Action是否有需要的操作。
3. Resource是否是需要的操作对象。
4. Effect是否是Allow而不是Deny。
5. Condition是否正确。

如果检查无法发现错误，需要调试步骤如下：

1. 如果有Condition的话先将其去掉。
2. Effect中去除Deny。
3. Resource换成“Resource”：“*”。
4. Action换成“Action”：“oss:*”。

> **说明**
>
> - 权限策略的生成推荐使用OSS授权策略生成工具RAM Policy Editor。
> - 如果想更多了解阿里云访问控制（RAM），请参见阿里云访问控制初探 。

4.OSS防盗链（Referer）配置及错误排除

本文介绍OSS防盗链配置及常见错误排查方法。

什么是Referer

`Referer`是HTTP Header的一部分，当浏览器向Web服务器发送请求时，一般会带上Referer，告诉服务器从哪个页面链接过来的。

Referer的作用

- 防盗链。比如，网站访问自己的图片服务器，图片服务器取到Referer来判断是不是自己的域名，如果是就继续访问，不是则拦截。
- 数据统计。比如，统计用户是从哪里链接过来访问的。

Referer为空

空Referer指的是HTTP请求中Referer头部的内容为空，或者HTTP请求中不包含Referer头部。

下面两种情况Referer为空：

- 当请求并不是由链接触发产生。比如，直接把地址输入地址栏里打开页面。
- 从https页面上的链接访问到非加密的http页面时，在http页面上是检查不到Referer的。

在防盗链设置中，允许空Referer和不允许空Referer有什么区别呢？

- 在防盗链的白名单设置中，如果指明白名单中包含空的Referer，那么通过浏览器地址栏直接访问该资源URL是可以访问到的。
- 如果不指名需要包含空的Referer，那么通过浏览器直接访问也是被禁止的。

OSS防盗链

OSS防盗链是通过`Referer`来实现的，所以也简称为`Refer`或`refer`，详细说明请参见OSS防盗链。

- OSS防盗链配置

  OSS防盗链包括：

  - 是否允许Referer字段为空的请求访问
  - Referer字段白名单

  OSS防盗链通过在控制台或SDK设置bucket属性来配置

- OSS防盗链注意点

  OSS防盗链配置中有以下注意点：

  - 用户只有通过URL签名或者匿名访问Object时，才会做防盗链验证。请求Header中有`Authorization`字段时，不会做防盗链验证。
  - 一个Bucket可以支持多个Referer参数，这些参数之间由`,`分隔。
  - Referer参数支持通配符`*`和`?`，详细解释参见下面的通配符说明。
  - 用户可以设置`是否允许Referer字段为空`的请求访问。
  - 白名单为空时，不会检查Referer字段是否为空（不然所有的请求都会被拒绝）。

- 白名单不为空，且设置了不允许Referer字段为空的规则，则只有Referer属于白名单的请求被允许，其它请求（包括Referer为空的请求）会被拒绝。
- 如果白名单不为空，但设置了允许Referer字段为空的规则，则Referer为空的请求和符合白名单的请求会被允许，其它请求都会被拒绝。
- Bucket的三种权限（private、public-read、public-read-write）都会检查Referer字段。

通配符说明：

- 星号 `*` ：代替0个或多个字符。如果正在查找以AEW开头的一个文件，但不记得文件名其余部分，可以输入AEW*，查找以AEW开头的所有文件类型的文件，如AEWT.txt、AEWU.EXE、AEWI.dll等。要缩小范围可以输入AEW*.txt，查找以AEW开头，并以.txt为扩展名的文件，如AEWIP.txt、AEWDF.txt。
- 问号 `?` ：代替一个字符。如果输入love?，查找以love开头的一个字符结尾文件类型的文件，如lovey、lovei等。要缩小范围可以输入love?.doc，查找以love开头，任意一个字符结尾，并以.doc为扩展名的文件，如lovey.doc、loveh.doc。

- 典型配置
  - 所有请求都可以访问
    - 空Refer：允许refer为空
    - Refer列表： 空
  - 带规定的Referer的请求才能访问
    - 空Refer：不允许refer为空
    - Refer列表： `http://*.oss-cn-beijing.aliyuncs.com` ， `http://*.aliyun.com`
  - 带规定的Referer的请求、不带Referer的请求可以访问
    - 空Refer：允许refer为空
    - Refer列表： `http://*.oss-cn-beijing.aliyuncs.com` ， `http://*.aliyun.com`

常见错误及排除

当Referer配置错误时，HTTP状态码（http code）是403，OSS返回如下错误：

```
<Code>AccessDenied</Code>
<Message>You are denied by bucket referer policy.</Message>
```

> **说明**
> - Referer报错一般是站点类应用，浏览器中可以查看Header的Referer。如Chrome，按 `F12` 打开 `开发者工具` ，在 `Network` 中查看相应元素的Header。
> - OSS返回的错误可以通过抓包获取。如Wireshark，筛选器可以指定为 `host bucket-name.oss-cn-beijing.aliyuncs.com` 。

可能的原因：

- Referer为空，请求Header中没有Referer字段或者Referer字段为空。
- Referer不在规定的Referer范围内。以下几点请注意：
  - `http://` 还是 `https://` 配置时请确认。
  - a.aliyun.com和b.aliyun.com，匹配于 `http://*.aliyun.com` 或 `http://?.aliyun.com` ；

- domain.com匹配于 `http://domain.com` ，而不是 `http://*.domain.com` ；

- Referer格式错误，Refer配置必须带 `http://` 或者 `https://` ，否则无效。如b.aliyun.com是无效配置。

> 说明
> - 在OSS控制台 > Bucket > Bucket属性 > 防盗链 中配置Referer。
> - 调试时请清空浏览器缓存。
> - OSS的Refer只支持白名单， 暂时不支持黑名单。

其它错误的排除请参看OSS 403错误及排除。

其它问题

设置防盗链后，为什么用curl还是能抓到视频？

- 检查是否开启了CDN。CDN的Refer设置不能为空，且防盗链名单需与OSS一致。
- 调试OSS的Referer时，请先排除CDN的影响。先调好OSS的Referer，再调CDN的Referer。

> 说明 Referer更详细的介绍及配置请参见防盗链。

5.上传回调错误及排除

本文主要介绍上传回调中的常见错误及分析处理。

关于上传回调

OSS在文件上传完成时可以提供回调（Callback）给用户的回调服务器（Callback Server）。在上传请求中携带相应的回调参数，即能实现上传回调。支持上传回调的 API 接口有：PutObject、PostObject、CompleteMultipartUpload。更详细的介绍请参见开发指南中的上传回调和Callback API。

> **说明** 回调服务器（Callback Server），有时也叫业务服务器。

应用场景

- 通知。上传回调的一种典型应用是授权的第三方上传文件时指定回调参数。上传完成后OSS向回调服务器发送回调请求。回调服务器收到回调请求后，记录上传信息。
- 处理、审查、统计等。回调服务器收到回调请求后，对上传的文件做处理、审查、统计等。

数据流

OSS上传回调流程如下图：

OSS上传回调流程解释如下：

数据流	含义	说明
1	上传文件并携带回调参数。格式请参见PostObject。	通过SDK调用API接口（PutObject、CompleteMultipartUpload、PostObject）完成。
2	OSS存储文件后发起回调	OSS向上传请求中指定的CallbackUrl 发起POST请求，回调超时时间是5秒。 超时时间为固定值，不支持配置。 回调请求POST的格式请参见发起回调请求。
3	回调服务器返回处理结果	• 回调服务器返回的消息体一定要是 Json 格式。 • OSS认为非200请求为回调失败。参数无效、回调失败返回 `40x` ；超时、无法连接返回 `50x` 。
4	OSS返回上传、回调结果	• 上传、回调都成功，返回 `200` 。 • 上传成功、回调失败返回 `203` ，ErrorCode为 `CallbackFailed` ，ErrorMessage描述错误原因。

SDK/PostObject

上传时可以通过设置回调参数，指定回调服务器URL、发送给回调服务器的数据、格式等。回调服务器处理回调时，需要一些上下文信息，如 `bucket` 、 `object` 等，通过系统变量指定；系统变量以外的上下文信息，通过自定义变量指定。

上传回调包括以下参数：

字段	含义	说明
callbackUrl	回调服务器地址	必选参数
callbackHost	回调请求消息头中 `Host` 的值	可选参数，默认为 `callbackUrl`
callbackBody	回调请求的消息体	必选参数，内容可以包括系统变量和自定义变量
callbackBodyType	回调请求消息头中 `Content-Type` 的值，即 `callbackBody` 的数据格式	可选参数，支持 `application/x-www-form-urlencoded` 和 `application/json` ，默认为前者

通过上传请求携带上传回调参数有两种实现方式：

- 通过消息头中的 `x-oss-callback` ，携带回调参数。这种方式比较常用，推荐该方式。
- 通过QueryString的 `callback` ，携带回调参数。

`x-oss-callback` 或 `callback` 的值生成规则如下：

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

`CallbackJson` 的值，示例如下：

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

或

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

系统变量及自定义变量

`CallbackJson` 示例中的 `callbackBody` 包括如 `${bucket}` 、 `${object}` 、 `${size}` 的变量，即为OSS定义的系统变量，OSS回调时会用实际值替换掉系统变量。OSS定义的系统变量如下表：

变量	含义
${bucket}	存储空间名称
${object}	文件名称
${etag}	文件的ETag
${size}	文件大小
${mimeType}	文件类型，如image/jpeg等
${imageInfo.height}	图片高度
${imageInfo.width}	图片宽度
${imageInfo.format}	图片格式，如jpg、png等

> **说明**
>
> - 系统变量大小写敏感。
> - 系统变量的格式为 `${bucket}` 。
> - imageInfo针对于图片格式，非图片格式值为空。

`CallbackJson` 示例中的 `callbackBody` 包括如 `${x:my_var}` 的变量，即自定义变量，OSS回调时会用自定义的值替换掉自定义变量。自定义变量的值可以在上传请求中定义并携带，有以下两种方式：

- 通过消息头中的 `x-oss-callback-var` ，携带自定义变量。这种方式比较常用，也是推荐方式。
- 通过QueryString的 `callback-var` ，携带自定义变量。

`x-oss-callback-var` 或 `callback-var` 的生成规则如下：

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

`CallbackVarJson` 值的示例如下：

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

> **说明**
>
> - 自定义变量必须以x:开头，大小写敏感，格式为 `${x:my_var}` 。
> - 自定义变量的长度受消息头、URL的长度限制，建议自定义变量不超过10个，总长度不超过512Byte。

SDK使用示例

部分SDK对上述步骤进行了封装，如Java、JS，部分SDK需要使用上面的规则生成上传回调参数和自定义变量，如Python、PHP、C。SDK的使用示例如下：

SDK	上传回调示例	说明
Java	CallbackSample.java	注意 `CallbackBody` 中的转义字符。
Python	object_callback.py	-
PHP	Callback.php	上传的$options中 `OSS_CALLBACK` 和 `OSS_CALLBACK_VAR` 不需要base64，SDK会处理。
C#	UploadCallbackSample.cs	使用 *using* 读取 `PutObjectResult.ResponseStream` ，但要确保关闭 `PutObjectResult.ResponseStream` 。
JS	object.test.js	-
C	oss_callback_sample.c	-
Ruby	callback.rb	-
iOS	上传后回调通知	`<var1>` 的格式为 `x:var1` 。
Andriod	上传后回调通知	注意 `CallbackBody` 中的转义字符。

> **说明** Go SDK暂不支持上传回调。

PostObject使用示例

PostObject支持上传回调，回调参数通过表单域 `callback` 携带，自定义变量通过独立的表单域携带，详情请参见PostObjet。

PostObject的使用示例如下：

SDK	上传回调示例
Java	PostObjectSample.java
Python	object_post.py
JS	JavaScript客户端签名直传
C#	PostPolicySample.cs

回调服务器

回调服务器（Callback Server），是一个HTTP服务器，处理OSS发送的回调请求，POST消息。回调服务器的URL即上传回调参数中的 `callbackUrl` 。回调服务器是用户自己实现的处理逻辑，实现上传数据的记录、审查、处理、统计等。

- 回调签名

  回调服务器为了确认收到的POST请求来自于OSS的上传回调，需要验证该POST消息的签名。回调服务器也可以不验证签名，直接处理该消息。为了提高回调服务器的安全性，建议验证消息签名。回调签名规则请参见回调签名。

  > **说明** OSS的回调服务器示例中提供了签名校验的实现，推荐直接使用该部分代码。

- 消息处理

  回调服务器的主要逻辑，对OSS的回调请求进行处理。以下几点请注意：

  - 回调服务器必须处理OSS的POST请求；
  - OSS回调的超时时间是 5秒，回调服务器必须在5秒内完成处理并返回；
  - 回调服务器返回给OSS的消息体必须是JSON格式；
  - 回调服务器是用户自己的逻辑，OSS提供示例而不提供具体业务逻辑实现。

- 实现示例

  回调服务器的实现示例如下：

语言	示例	运行方法
Java	AppCallbackServer.zip	解压后执行 `java -jar oss-callback-server-demo.jar 9000` 。
PHP	callback-php-demo.zip	Apache环境下部署运行。
Python	callback_app_server.py.zip	解压后执行 `python callback_app_server.py` 。
Ruby	oss-callback-server	执行 `ruby aliyun_oss_callback_server.rb` 。

调试步骤

上传回调的调试分为两部分：上传的客户端、处理回调的回调服务器。建议先调试客户端上传部分，再调试回调服务器部分。两部分单独调试完成后，再运行完整的上传回调。

- 调试客户端

  客户端调试时，可使用OSS提供的回调服务器 `http://oss-demo.aliyuncs.com:23450` ，即回调参数 `callbackUrl` 。该回调服务器只验证回调请求的签名，对回调请求不做处理。对于签名验证成功的回调请求，返回 `{"Status":"OK"}` ；签名验证失败的回调请求，返回 `400 Bad Request` ；非POST请求返回 `501 Unsupported method` 。示例回调服务器的代码请参见callback_app_server.py.zip。

- 调试回调服务器

  回调服务器是一个支持处理POST请求的HTTP服务器，可以在OSS提供的示例基础上修改，也可以自己独立实现。OSS提供的回调服务器示例：

语言	示例	运行方法
Java	AppCallbackServer.zip	解压后执行 `java -jar oss-callback-server-demo.jar 9000` 。
PHP	callback-php-demo.zip	Apache环境下部署运行。
Python	callback_app_server.py.zip	解压后执行 `python callback_app_server.py` 。
C#	callback-server-dotnet.zip	编译后执行 `aliyun-oss-net-callback-server.exe 127.0.0.1 80` 。
Go	callback-server-go.zip	编译后执行 `aliyun_oss_callback_server` 。
Ruby	oss-callback-server	执行 `ruby aliyun_oss_callback_server.rb` 。

回调服务器可以通过cURL命令调试，下面几个命令可能会用到：

```
# 向回调服务器发送消息体为 `object=test_obj` 的 `POST` 请求，可以使用如下命令
curl -d "object=test_obj"  http://oss-demo.aliyuncs.com:23450 -v
# 向回调服务器发送消息体为文件 `post.txt` 内容 的 `POST` 请求，可以使用如下命令
curl -d @post.txt  http://oss-demo.aliyuncs.com:23450 -v
#向回调服务器发送消息体为文件 `post.txt` 内容的 `POST` 请求，并携带指定的消息头 `Content-Type`
curl -d @post.txt -H "Content-Type: application/json" http://oss-demo.aliyuncs.com:23450
-v
```

> **说明**
>
> - 调试回调服务器时，可以先忽略签名验证部分，因为 `cURL` 模拟签名功能比较困难。
> - 签名验证功能OSS示例中已经提供，建议直接使用。
> - 回调服务器建议有日志功能，记录收到的所有消息，方便调试、跟踪。
> - 回调服务器正确处理回调请求后，一定要返回200，而不是其它的 `20x` 。
> - 回调服务器返回给OSS的消息体，一定要是JSON格式， `Content-Type` 设置为 `application/json` 。

常见错误及原因

- 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":${bucket},"objec
t":${object}}","callbackBodyType":"application/json"}</ArgumentValue>
</Error>
```

说明 回调参数设置错误，或参数格式错误。常见的错误是 `ArgumentValue` 之间的回调参数，不是有效JSON格式。在 JSON中 `\` 、 `"` 是转义字符，如 `"callbackBody":"{"bucket":${bucket},"object":${object}}"` 应该为 `"callbackBody":"{\"bucket\":${bucket},\"object\":${object}}"` 。针对具体的SDK，请参见对应的上传回调示例，详细请参考SDK使用示例部分。

转义后的字符	转义前的字符
\\	\\\\
\"	\\\"
\b	\\b
\f	\\f
\n	\\n
\r	\\r
\t	\\t

- CallbackFailed

  CallbackFailed 常见示例如下：

  - 示例1：

```
<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>
```

说明 回调服务器返回给OSS的消息体非JSON格式。您可以通过 `curl -d "<Content>" <CallbackServerURL> -v` 或抓包确认内容。Windows下推荐使用工具Wireshark 抓包，Linux下使用命令tcpdump抓包。一些非法返回消息体如下： `OK` ， `\357\273\277{"Status":"OK"}` （即含有 `ef bb bf` 三个字节的BOM头）等。

- 示例2：

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

> **说明** 回调服务器处理时间超过5秒，OSS认为超时。建议回调服务器的处理逻辑修改为异步，保证在5秒内处理完毕并返回结果给OSS。

- 示例3：

```
<Error>
  <Code>CallbackFailed</Code>
  <Message>Error status : -1 8.8.8.8:9090 reply timeout, cost:5000ms, timeout:5000ms (e
rr -4, errno115)</Message>
  <RequestId>587C8D382AE0B92FA3EEF62C</RequestId>
  <HostId>bucket.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
```

> **说明** 回调服务器处理时间超过5秒，OSS认为超时。

- 示例4：

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

> **说明** 回调服务器返回给OSS的消息的状态码是 `400` ，请检查回调服务器的处理逻辑。

- 示例5：

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

> **说明** 回调服务器未启动，或者缺少上传回调参数中的 `CallbackUrl` ，或者OSS与回调服务器的网络不通。推荐在ECS上部署回调服务器，与OSS同属内网可以节省流量费用，同时保证网络质量。

- 返回的Body非JSON格式

示例如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>CallbackFailed</Code>
  <Message>Response body is not valid json format.</Message>
  <RequestId>5763E6EC338A486CBF8B544A</RequestId>
  <HostId>guoping-hangzhou.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
```

错误原因有以下两种情况。

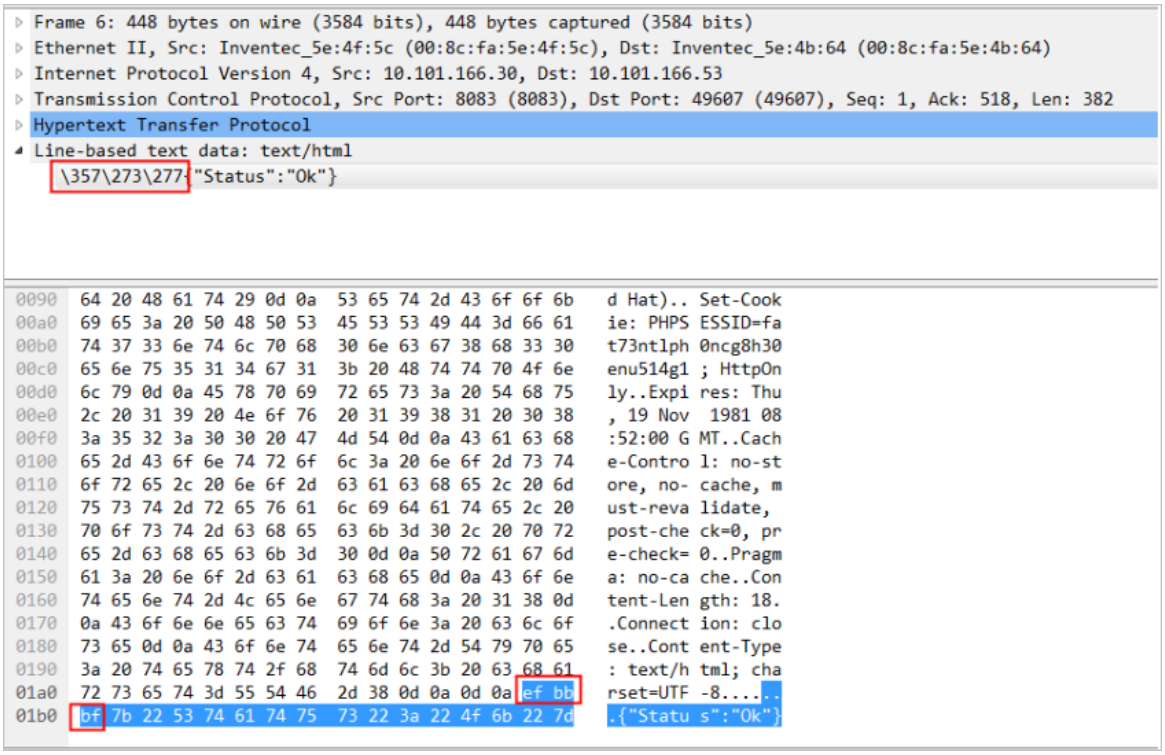
- 应用服务器返回给OSS的Body不是JSON格式。如下图所示：

```
#response to OSS
resp_body = '{"Status":"OK"'
self.send_response(200)
self.send_header('Content-Type', 'application/json')
self.send_header('Content-Length', str(len(resp_body)))
self.end_headers()
self.wfile.write(resp_body)
```

Resp_body不是合法的JSON格式，OSS就会报上述错误。这种一般比较明显，还有比较隐蔽的。比如应用服务器处理过程中抛出异常，导致没有按照预期返回给OSS，而是返回了一些栈信息等。

- 应用服务器返回给OSS的Body中带有bom头。

这类错误常见于用php编写的应用服务器中。由于php返回了bom头，导致OSS收到的Body中多了三个字节，不符合JSON格式，因此报上述错误。如果在应用服务器端抓包，可以看到以下信息。



上图中ef bb bf这三个字节就是bom头。

> **说明** 应用服务器返回OSS响应时，请去掉bom头。

- 错误的status

此类错误较多，比如502、400等。示例如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>CallbackFailed</Code>
  <Message>Error status : 502.</Message>
  <RequestId>5763F3C8B47FB809EB5700DA</RequestId>
  <HostId>guoping-hangzhou.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
```

> 说明
>
> - 400或者其他的status比如404/403等是指应用服务器返回给OSS的HTTP status是400或者404/403等，正常情况下应用服务器必须返回200给OSS。
> - 502是由于应用服务器根本就没有开启web服务，没有监听OSS发过来的回调请求。

- 超时

  示例如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>CallbackFailed</Code>
  <Message>Error status : -1.call 10.157.223.86:8080  reply timeout, cost:5000ms, timeout:5000ms (err:-4, errno:115)</Message>
  <RequestId>5763F580727EC3D8022E14B6</RequestId>
  <HostId>guoping-hangzhou.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
```

> 说明 出于安全考虑，OSS的回调请求只会等待5秒。如果5秒后还没有返回，那么OSS就会主动断开与应用服务器的连接，并返回给客户端超时错误，错误信息中的IP可以忽略。

6.STS常见问题及排查

本文介绍STS AssumeRole（角色扮演）常见错误及错误原因。

角色扮演常见错误及原因如下：

序号	错误	原因
1	ErrorCode: NoPermission ErrorMessage: Roles may not be assumed by root accounts.	使用主用户的密钥调用AssumeRole，请使用子用户的密钥。
2	ErrorCode: MissingSecurityToken ErrorMessage: SecurityToken is mandatory for this action.	使用临时用户的密钥调用AssumeRole，请使用子用户的密钥。
3	Error code: InvalidAccessKeyId.NotFound Error message: Specified access key is not found	AccessKeyId无效，请检查是否写错，特别是前后不能有空格。
4	Error code: InvalidAccessKeyId.Inactive Error message: Specified access key is disabled.	使用的子用户的密钥，已经被禁止，请启用密钥或更换密钥。 密钥是否被禁止，请通过控制台的**访问控制 > 用户管理 > 管理 > 用户详情 > 用户AccessKey**确认，并开启。
5	ErrorCode: InvalidParameter.PolicyGrammar ErrorMessage: The parameter Policy has not passed grammar check.	角色扮演时指定的授权策略无效。角色扮演时可以指定授权（Policy），也可以不指定。如果指定授权策略，则临时用户的权限是指定的授权策略和角色权限的交集；如果不指定授权策略，临时用户的权限是角色的权限。报该错误时，请检查指定的授权策略Policy。不推荐临时用户扮演角色时指定授权策略。如果的确需要使用授权策略，强烈建议使用RAM Policy Editor生成授权策略。
6	ErrorCode: InvalidParameter.RoleSessionNameErrorMessage: The parameter RoleSessionName is wrongly formed.	角色扮演时指定RoleSessionName无效。此参数用来区分不同的Token，以标明谁在使用此Token，便于审计。格式：^[a-zA-Z0-9.@-_]+$，2-32个字符。 了解更多请参看扮演角色操作接口。 如命名 `a，1，abc\*abc，忍者神龟` 等都是非法的。

序号	错误	原因
7	ErrorCode: InvalidParameter.DurationSeconds Error message: The Min/Max value of DurationSeconds is 15min/1hr.	角色扮演时指定的过期时间无效，即AssumeRoleRequest.setDurationSeconds参数无效。角色扮演时可以指定过期时间，单位为秒，有效时间是900 ~ 3600，如 `assumeRoleRequest.setDurationSeconds(60L * 20)` ，20分钟内有效。
8	ErrorCode: NoPermissionErrorMessage: No permission perform sts:AssumeRole on this Role. Maybe you are not authorized to perform sts:AssumeRole or the specified role does not trust you	• 原因1：AssumeRole的子用户没有权限，请给子用户授予AliyunSTSAssumeRoleAccess系统授权策略。请在**访问控制 > 用户管理 > 授权 > 可授权策略名称**中给子用户授权AliyunSTSAssumeRoleAccess。 • 原因2：申请角色扮演的子用户的云账号ID与角色的“受信云账号ID”不符，请角色创建者确认并修改。子用户的云账号ID，即创建子用户的主用户的ID；角色的云账号ID，即创建角色的主用户的云账号ID。请在如下位置确认修改：**访问控制 > 角色管理 > 管理 > 角色详情 > 编辑基本信息**中确认修改。 • 原因3：角色的类型错误，如果角色的类型分为“用户角色”和“服务角色”，服务角色不能使用AssumeRole扮演临时用户。

说明

- Java角色扮演的示例，请参看GitHub。
- 其它的原因的角色扮演示例，请参考STS SDK使用手册。
- 您想更多了解阿里云访问控制（RAM），请参看阿里云访问控制初探 。

7.OSS跨域资源共享（CORS）错误及排除

本文介绍跨域资源共享（Cross Origin Resource Sharing，简称 CORS）配置常见错误及解决方案。

CORS的介绍及配置请参看跨域资源共享最佳实践。

配置项

CORS配置有以下几项：

- 来源（AllowedOrigin）

  允许跨域请求的来源，可以同时指定多个。配置时需带上完整的域信息，例如 `http://10.100.100.100:8001` 或 `https://www.aliyun.com` 。注意，不要遗漏了协议名http或https，如果端口不是默认的 `80` ，还需要带上端口。如果不能确定域名，可以打开浏览器的调试功能，查看header中的 `Origin` 。域名支持通配符 `*` ，每个域名中允许最多使用一个 `*` ，例如 `https://*.aliyun.com` 。如果来源指定为 `*` ，则表示允许所有来源的跨域请求。

- Method

  按照需求开通对应的方法即可，调试时可以全部选择。

- Allow Header

  允许的跨域请求header。允许配置多条匹配规则，以回车间隔。在Access-Control-Request-Headers中指定的每个header，都必须在Allowed Header中有对应项。Header容易遗漏，没有特殊需求的情况下，建议设置为 `*` ，表示允许所有。大小写不敏感。

- Expose Header

  暴露给浏览器的header列表，即用户从应用程序中访问的响应头（例如一个JavaScript的XMLHttpRequest对象）。不允许使用通配符。具体的配置需要根据应用的需求确定，只暴露需要使用的header。如果不需要暴露可以不填。大小写不敏感。该项是可选配置项。

- 缓存时间（MaxAgeSeconds）

  浏览器对特定资源的预取请求（OPTIONS请求）返回结果的缓存时间，单位为秒。如果没有特殊情况可以稍大一点，比如60秒。该项是可选配置项。

CORS的配置方法一般是针对每个访问来源单独配置规则，不将多个来源混到一个规则，多个规则之间不要有覆盖冲突。其它的选项只开放需要的权限即可。

错误排除

报错

CORS配置错误会报如下错误：

- 浏览器报类似如下错误：

```
OPTIONS http://bucket.oss-cn-beijing.aliyuncs.com/
XMLHttpRequest cannot load http://bucket.oss-cn-beijing.aliyuncs.com/. Response to prefli
ght request doesn't pass access control check: No 'Access-Control-Allow-Origin' header is
present on the requested resource. Origin '{yourwebsiet}' is therefore not allowed access
. The response had HTTP status code 403.
```

- OSS报如下错误：

```
<Code>AccessForbidden</Code>
<Message>CORSResponse: This CORS request is not allowed. This is usually because the evaluation of Origin, request method / Access-Control-Request-Method or Access-Control-Requet-Headers are not whitelisted by the resource's CORS spec.</Message>
```

说明

- CORS报错一般是站点类应用导致，浏览器中可以查看请求详情。如Chrome，按 `F12` 打开 `开发者工具` ，在 `Network` 中查看相应元素。
- OSS返回的错误可以通过抓包获取。如使用Wireshark，筛选器可以指定为 `host bucket-name.oss-cn-beijing.aliyuncs.com` 。
- OSS返回的错误也可以通过CORS的调试程序oss-h5-upload-js-direct界面提示获取。

其它错误请参看OSS 403错误及排除。

排错

CORS可能错误如下：

- 来源（AllowedOrigin）配置不正确
- Method（AllowedMethod）配置错误
- Allow Header 配置错误
- Expose Header 配置错误

调试方法：

- 将 来源（AllowedOrigin）设置成 `*` ，确认该配置项无误。如果设置成 `*` 后可以成功上传，说明是来源（AllowedOrigin）配置错误，请根据规则认真检查。
- 选择 Method（AllowedMethod） 的全部选项（GET、PUT、DELETE、POST、HEAD），确认该配置项目无误。
- 将 Allow Header 配置成 `*` ，确认该配置无误。
- 将 Expose Header 设置为指定值或者不填，确认该项配置无误。

说明 在OSS控制台，选择Bucket后，通过**Bucket属性 > 跨域设置**配置上述选项。

8.OSS 403错误及排查

本文介绍访问OSS时出现403错误的常见原因及解决方案。

UserDisable.UserDisable错误

当您访问OSS遇到如下的UserDisable错误：

```
<Code>UserDisable</Code>
<Message>UserDisable</Message>
```

有以下两类原因：

- 欠费被禁

  确认欠费方法：在OSS 管理控制台上打开**费用中心**，检查是否欠费。如果欠费，请及时充值。

  > **说明**
  > - OSS欠费后，还可以正常使用24小时，24小时后禁止访问。
  > - 历史数据保留15天，15天后历史数据将被删除。
  > - 当您在消息中心看到阿里云OSS欠费提醒后，请及时充值，否则会影响您的正常使用。

- 安全原因被禁

  在OSS 管理控制台上打开**消息中心**，在右侧的**安全消息**中查看违规通知。违规的原因有很多，例如您使用OSS做私服，您的图片涉黄、涉暴等。

  > **说明** 如果您有账户处于被禁状态，请务必处理，重新申请新账户，无法保证正常使用。

RequestTimeTooSkewed.The difference between...错误

访问OSS遇到如下的RequestTimeTooSkewed错误：

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

原因：您发送请求的时间与OSS收到请求的时间，间隔超出了15分钟，OSS从安全考虑认为该请求是无效的，返回上述错误。请检查发送请求设备的系统时间，并根据时区调整到正确时间。

您可能会有下面的疑问：

- 发送请求的机器或设备的系统时间，调整标准是什么呢？

  OSS的系统时间采用GMT时间，您的设备的系统时间，需要调整到GMT时间，或与其相对应的时区时间。GMT（Greenwich Mean Time）是零时区的区时，即世界标准时间。

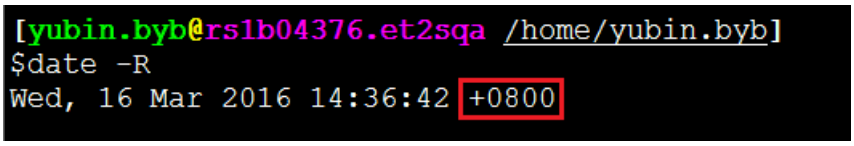
  例如，您访问OSS的设备系统配置是东八区，系统时间调整到比GMT早8小时。我国的标准时间—北京时间—就是东八区时间。如果您的系统时间是东八区，那么您的系统时间调整到北京时间即可。

  - Windows系统查看时区的方法：

    通过**控制面板 > 时钟、语言和区域 > 设置日期和时间**，打开日期和时间，时区栏的+08:00表示您的设备时区是东八区。

- Linux/Unix系统查看时区的方法：

  请执行 `date -R` 查看时间和时区。下图中的 +0800，表示您的设备系统时区是东八区。



- 使用多个地域的OSS，例如杭州、新加坡、美国，时间同步有问题吗？

  没有问题。每个地域的OSS都使用GMT时间，您发送请求的设备系统时间也是GMT时间。

InvalidAccessKeyId.The OSS Access Key Id...错误

访问OSS遇到如下的错误：

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

原因：您的AccessKeyID禁用或不存在。排查方法如下：

登录阿里云控制台的 AccessKey 管理，确认访问OSS使用的AccessKeyID存在且处于启用状态。

- 如果您的AccessKeyID处于禁用状态，请开启。
- 如果您的AccessKeyID不存在请创建，并使用新的AccessKeyID访问OSS。

AccessDenied.The bucket you are attempting to...错误

访问OSS遇到如下的错误：

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

原因：您访问Bucket使用的Endpoint不正确，如果您需要了解Endpoint的详细信息，请参看OSS 基本概念。

怎么找到正确的Endpoint呢？

如果SDK异常抛出如下的异常，或返回如下错误：

```
<Error>
  <Code>AccessDenied</Code>
  <Message>The bucket you are attempting to access must be addressed using the specified en
dpoint. 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>
```

- 其中 `Endpoint` 中的 `oss-cn-****.aliyuncs.com` 就是正确的Endpoint，请使用 `http://oss-cn-****.aliyuncs.com` 或 `https://oss-cn-****.aliyuncs.com` 作为Endpoint访问OSS。
- 如果错误中没有 `Endpoint`，请登录OSS控制台，在**Bucket管理**中找到您访问的Bucket，单击进入**Bucket概览**页面。OSS域名中可以看到内网和外网域名。

- 外网域名是在公网上访问OSS使用的域名；内网域名是指在阿里云内部访问的OSS使用的域名。例如您在您的ECS上访问OSS，可以使用内网域名。
- Endpoint是域名去掉Bucket部分，加上访问协议。例如上图中OSS的公网域名是 `oss-****.aliyuncs.com` ，它的公网Endpoint是 `http://oss-cn-****.aliyuncs.com` ；类似，内网Endpoint是 `http://oss-cn-****-internal.aliyuncs.com` 。

ImageDamage.The image file may be damaged错误

访问OSS遇到如下的错误：

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

原因：说明图片文件有部分信息丢失或损坏，导致无法正常识别或处理。您可能会有一个疑问，某图片在本地用图片处理器可以打开，但是使用OSS处理时报错。原因是，图片浏览器会对损坏的图片做些处理，OSS图片服务暂时没有这个操作。

AccessDenied.AccessDenied错误

访问OSS遇到如下的错误：

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

原因：说明访问OSS的用户没有当前操作的权限。请确认使用的 `AccessKeyID/AccessKeySecret` 是正确的。如果使用的是子帐号/临时账户（STS），请确认当前用户的权限。确认方法：

在访问控制管理控制台单击**用户管理**，单击需要确认权限的用户，单击**用户授权策略 > 加入组的授权策略**查看该用户的权限，确认是否已经赋予当前用户Bucket/Object的操作权限。

SignatureDoesNotMatch.The request signature we calculated...错误

访问OSS遇到如下的错误：

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

请按照以下步骤排查：

1. 检查endpoint

   请检查endpoint前面没有Bucket，后面没有多余的 `/` ，前后没有多余的 `空格` 。例如下面的endpoint均不合法： `http://my-bucket.oss-cn-hangzhou.aliyuncs.com` 、 `http://oss-cn-hangzhou.aliyuncs.com/` 、 `http:// oss-cn-hangzhou.aliyuncs.com` 以及 `https:// oss-cn-hangzhou.aliyuncs.com` 。

2. 检查AccessKeyID/AccessKeySecret

   请确认AccessKeyID/AccessKeySecret正确，确保AccessKeyID/AccessKeySecret前后都没有空格，特别是使用了复制粘贴的情况。

3. 检查Bucket Name/Object Key

   请确保Bucket Name/Object Key命名合法有效，符合要求。

- Bucket命名规范：只能包括小写字母、数字和短横线（-），必须以小写字母或者数字开头，长度必须在3-63字节之间。
- Object的命名规范：使用UTF-8编码，长度必须在1-1023字节之间，不能以“/”或者“\”字符开头。

4. 如果是您自己实现的签名，请使用OSS SDK提供的签名方法。

   OSS SDK提供了URL/Header签名的实现，详细请参看授权访问。

5. 如果您的环境不适合使用SDK，需要自己实现签名，签名方法请参考用户签名验证，仔细检查每个签名字段。

   OSS的论坛上提供了一个可视化签名的工具，请比较每个签名字段和最后的签名，签名工具地址。

6. 如果您使用了代理，请检查代理服务器是否添加额外的Header。

其它错误

请根据SDK返回的错误码、错误信息判断原因，特别是错误信息会提示错误原因。如果怀疑错误跟网络环境有关，请使用ossutil排查问题，ossutil会给出可能的原因。

9.自签名计算失败

通过 header 或者 URL 中自签名计算 signature 时，经常会遇到计算签名失败 “The request signature we calculated does not match the signature you provided” 。本文 demo 演示了如何调用 API 自签名时上传 Object 到 OSS。

使用方法

```
[root@izw4z home]# python PutObject.py -h
Usage: PutObject.py [options]
Options:
  -h, --help  展示帮助文档
  -i AK       客户控制台上云账号或者 RAM 子账号的 Accesskey ，必选项
  -k SK       客户控制台上云账号或者 RAM 子账号的 AccessKeySecrety ，必选项
  -e ED       OSS endpoint 地址，OSS 控制台概述上可以看到，必选项
  -b BK       OSS bucket 名称，必选项
  -o OBJECTS  上传到 OSS 的 object 名称，必选项
  -f FI       读取本地文件的路经，必选项
python PutObject.py -i B0g3mdt -k lNCA4L0P43Ax -e oss-cn-shanghai.aliyuncs.com -b yourBucke
t -f localfile.txt -o aliyun.txt
```

请求头

```
PUT /yuntest HTTP/1.1
Accept-Encoding: identity
Content-Length: 147
Connection: close
User-Agent: Python-urllib/2.7
Date: Sat, 22 Sep 2018 04:36:52 GMT
Host: yourBucket.oss-cn-shanghai.aliyuncs.com
Content-Type: application/x-www-form-urlencoded
Authorization: OSS B0g3mdt:lNCA4L0P43Ax
```

响应头

```
HTTP/1.1 200 OK
Server: AliyunOSS
Date: Sat, 22 Sep 2018 04:36:52 GMT
Content-Length: 0
Connection: close
x-oss-request-id: 5BA5C6E4059A3C2F
ETag: "D0CAA153941AAA1CBDA38AF"
x-oss-hash-crc64ecma: 8478734191999037841
Content-MD5: 0MqhU5QbIp3Ujqqhy9o4rw==
x-oss-server-time: 15
```

注意

- Signature 中所有加入计算的参数都要放在 header 中，保持 header 和 Signature 一致。
- PUT 上传时，Signature 计算的 Content-Type 必须是 application/x-www-form-urlencoded 。

- 通过 header 方式进行签名认证时无法设置过期时间。目前只有 SDK 、URL 签名支持设置过期时间。

源码

```
#! /us/bin/envy python
#Author: handle
#Update: 2018-09-29
from optparse import OptionParser
import urllib, urllib2
import datetime
import base64
import hmac
import sha
import os
import sys
import time
class Main():
  # Initial input parse
  def __init__(self,options):
    self.ak = options.ak
    self.sk = options.sk
    self.ed = options.ed
    self.bk = options.bk
    self.fi = options.fi
    self.oj = options.objects
    self.left = '\033[1;31;40m'
    self.right = '\033[0m'
    self.types = "application/x-www-form-urlencoded"
    self.url = 'http://{0}.{1}/{2}'.format(self.bk,self.ed,self.oj)
  # Check client input parse
  def CheckParse(self):
    if (self.ak and self.sk and self.ed and self.bk and self.oj and self.fi) != None:
      if str(self.ak and self.sk and self.ed and self.bk and self.oj and self.fi):
        self.PutObject()
    else:
      self.ConsoleLog("error","Input parameters cannot be empty")
  # GET local GMT time
  def GetGMT(self):
    SRM = datetime.datetime.utcnow()
    GMT = SRM.strftime('%a, %d %b %Y %H:%M:%S GMT')
    return GMT
  # GET Signature
  def GetSignature(self):
    mac = hmac.new("{0}".format(self.sk),"PUT\n\n{0}\n{1}\n/{2}/{3}".format(self.types,self
.GetGMT(),self.bk,self.oj), sha)
    Signature = base64.b64encode(mac.digest())
    return Signature
  # PutObject
  def PutObject(self):
    try:
      with open(self.fi) as fd:
        files = fd.read()
    except Exception as e:
      self.ConsoleLog("error",e)
```

```
    try:
      request = urllib2.Request(self.url, files)
      request.add_header('Host','{0}.{1}'.format(self.bk,self.ed))
      request.add_header('Date','{0}'.format(self.GetGMT()))
      request.add_header('Authorization','OSS {0}:{1}'.format(self.ak,self.GetSignature()))
      request.get_method = lambda:'PUT'
      response = urllib2.urlopen(request,timeout=10)
      fd.close()
      self.ConsoleLog(response.code,response.headers)
    except Exception,e:
      self.ConsoleLog("error",e)
  # output error log
  def ConsoleLog(self,level=None,mess=None):
    if level == "error":
      sys.exit('{0}[ERROR:]{1}{2}'.format(self.left,self.right,mess))
    else:
      sys.exit('\nHTTP/1.1 {0} OK\n{1}'.format(level,mess))
if __name__ == "__main__":
  parser = OptionParser()
  parser.add_option("-i",dest="ak",help="Must fill in Accesskey")
  parser.add_option("-k",dest="sk",help="Must fill in AccessKeySecrety")
  parser.add_option("-e",dest="ed",help="Must fill in endpoint")
  parser.add_option("-b",dest="bk",help="Must fill in bucket")
  parser.add_option("-o",dest="objects",help="File name uploaded to oss")
  parser.add_option("-f",dest="fi",help="Must fill localfile path")
  (options, args) = parser.parse_args()
  handler = Main(options)
  handler.CheckParse()
```

10.视频播放异常

使用背景

上传到 OSS 的视频有的可以在线播放，有的只有声音没有视频，使用此工具可以帮助您分析哪些浏览器支持在线播放，以及不能播放异常的原因。此工具需要 pip install ffprobe3 开源库。

使用方法

正常结果：

```
[root@izw4z home]# python AnalysisVideos.py test.mp4
[INFO:] Stream: 1
[INFO:]  Frame Rate:30.0
[INFO:]  Frame Size:(854, 480)
[INFO:]  Duration:116.5
[INFO:]  Frames:3495
[INFO:]  Is video: True
[INFO:]  video encode:h264
[CheckResult:]Chrom can playing video
[INFO:] Stream: 2
[INFO:]  Duration:116.561
[INFO:]  Frames:5465
[INFO:]  Is audio: True
[INFO:]  audio encode:aac
[CheckResult:]Chrom and FireFox can playing audio
```

异常情况：请参考报错信息

源码

```
#!/usr/bin/env python
#-*- coding:utf8 -*-
#Author: hanli
#Update: 2018-09-24
from __future__ import print_function
import os
import sys
import subprocess
from ffprobe3 import FFProbe
from ffprobe3.exceptions import FFProbeError
class MainFun():
  '''
  color
  '''
  def __init__(self):
    self.left = '\033[1;31;40m'
    self.gren = '\033[1;32;40m'
    self.right = '\033[0m'
    self.videos = sys.argv[1]
  def CheckModule(self):
    try:
      from ffprobe3 import FFProbe
```

```
      from ffprobe3.exceptions import FFProbeError
    except:
      self.ConsoleLog("Not install ffprobe3, please do 'pip install ffprobe3'","warn")
    return True
  '''
  checkvideo valid
  '''
  def CheckVideo(self):
    try:
      media = FFProbe(self.videos)
      for index, stream in enumerate(media.streams, 1):
        self.ConsoleLog('\tStream: {0}'.format(index))
        try:
            if stream.is_video():
                frame_rate = stream.frames() / stream.duration_seconds()
                self.ConsoleLog('\t\tFrame Rate:{0}'.format(frame_rate))
                self.ConsoleLog('\t\tFrame Size:{0}'.format(stream.frame_size()))
            self.ConsoleLog('\t\tDuration:{0}'.format(stream.duration_seconds()))
            self.ConsoleLog('\t\tFrames:{0}'.format(stream.frames()))
            if stream.is_video():
              self.ConsoleLog('\t\tIs video: True')
              self.ConsoleLog('\t\tvideo encode:{0}'.format(stream.codec()))
              self.CheckResult(stream.codec(),"video")
            if stream.is_audio():
              self.ConsoleLog('\t\tIs audio: True')
              self.ConsoleLog('\t\taudio encode:{0}'.format(stream.codec()))
              self.CheckResult(stream.codec(),"audio")
        except FFProbeError as e:
            self.ConsoleLog(e,"warn")
        except Exception as e:
            self.ConsoleLog(e,"warn")
      return True
    except Exception as e:
      self.ConsoleLog(e,"warn")
  '''
  check result
  '''
  def CheckResult(self,codec,types=None):
    if types == 'video':
      if codec.lower() in ['h264','h265','h263','vp9','vp8','theora']:
        self.ConsoleLog("Chrom can playing video","result")
      elif codec.lower() in ['h264','theora']:
        self.ConsoleLog("FireFox can playing video","result")
      else:
        self.ConsoleLog("Chrom and FireFox not Support video encode type","exec")
    if types == 'audio':
      if codec.lower() in ['vorbis','wmv','aac','mp3']:
        self.ConsoleLog("Chrom and FireFox can playing audio","result")
      else:
        self.ConsoleLog("Chrom and FireFox not Support audio enccode type","exec")
  '''
  output log
  '''
  def ConsoleLog(self,level,tag=None):
```

```
    if tag == "warn":
      sys.exit("{0}[ERROR:]{1}{2}".format(self.left,self.right,level))
    elif tag == "result":
      print("{0}[CheckResult:]{1}{2}".format(self.gren,self.right,level))
    if tag == "exce":
      print("{0}[ERROR:]{1}{2}".format(self.left,self.right,level))
    else:
      print("[INFO:]{2}".format(self.gren,self.right,level))
'''
Main input
'''
if __name__ == '__main__':
  if MainFun().CheckModule():
    if MainFun().CheckVideo() == None:
      sys.exit('\033[1;31;40m[ERROR:]\033[0mInput file is not video file')
```