# 上传回调

用户只需要在发送给OSS的请求中携带相应的Callback参数，即能实现回调。现在支持CallBack的API 接口有：PutObject、PostObject、CompleteMultipartUpload。

## 构造CallBack参数

CallBack参数是由一段经过base64编码的Json字串，用户关键需要指定请求回调的服务器

URL（callbackUrl）以及回调的内容（callbackBody）。详细的Json字段如下：

|  |  |  |
| --- | --- | --- |
| 字段 | 含义 |  |
| callbackUrl | * 文件上传成功后OSS向此url发送回调请求，请求方法为   POST，body为  callbackBody指定的内容。正常情况下，该url需要响应  “HTTP/1.1 200  OK”，body必须为JSON格式，响应头Content-Length必须为合法的值，且不超过3MB。   * 支持同时配置最多5个url，以   ”;”分割。OSS会依次发送请求直到第一个返回成功为止。 | 必选项 |
|  | * 如果没有配置或者值为空则认为没有配置callback。 * 支持HTTPS地址。 * 为了保证正确处理中文等情况   ，callbackUrl需做url编码处理，比如http://example.com/中文.php?key=value&中文名称  =中文值 需要编码成  http://example.com/%E4%B8 %AD%E6%96%87.php?key=v alue&%E4%B8%AD%E6%96 %87%E5%90%8D%E7%A7%  B0=%E4%B8%AD%E6%96%8  7%E5%80%BC |  |
| callbackHost | * 发起回调请求时Host头的值，只有在设置了callbackUrl时才有效。 * 如果没有配置   callbckHost，则会解析 callbackUrl中的url并将解析出的host填充到callbackHost中 | 可选项 |
| callbackBody | * 发起回调时请求body的值   ，例如  ：key=$(key)&etag=$(etag) &my\_var=$(x:my\_var)。   * 支持OSS系统变量、自定义变量和常量，支持的系统变量如下   表所示 。自定义变量的支持方  式在PutObject和  CompleteMultipart中是通过 callback-var来传递，在  PostObject中则是将各个变量通过表单域来传递。 | 必选项 |
| calbackBodyType | * 发起回调请求的ContentType，支持application/xwww-form-urlencoded和   application/json，默认为前者  。   * 如果为application/x-wwwform-urlencoded，则 callbackBody中的变量将会被经过url编码的值替换掉，如果   为application/json，则会按照 json格式替换其中的变量。 | 可选项 |

示例json串如下

{

"callbackUrl":"121.101.166.30/test.php",

"callbackHost":"oss-cn-hangzhou.aliyuncs.com",

"callbackBody":"{\"mimeType\":${mimeType},\"size\":${size}}",

"callbackBodyType":"application/json"

}

{

"callbackUrl":"121.43.113.8:23456/index.html",

"callbackBody":"bucket=${bucket}&object=${object}&etag=${etag}&size=${size}&mimeType=${mimeType}&image Info.height=${imageInfo.height}&imageInfo.width=${imageInfo.width}&imageInfo.format=${imageInfo.format}&m y\_var=${x:my\_var}"

}

其中callbackBody中可以设置的系统变量有，其中imageInfo针对于图片格式，如果为非图片格式都为空：

|  |  |
| --- | --- |
| 系统变量 | 含义 |
| bucket | bucket |
| object | object |
| etag | 文件的etag，即返回给用户的etag字段 |
| size | object大小，CompleteMultipartUpload时为整个object的大小 |
| mimeType | 资源类型，如jpeg图片的资源类型为image/jpeg |
| imageInfo.height | 图片高度 |
| imageInfo.width | 图片宽度 |
| imageInfo.format | 图片格式，如jpg、png等 |

## 自定义参数

用户可以通过callback-var参数来配置自定义参数。

自定义参数是一个Key-Value的Map，用户可以配置自己需要的参数到这个Map。在OSS发起POST回调请求的时候，会将这些参数和上一节所述的系统参数一起放在POST请求的body中以方便接收回调方获取。

构造自定义参数的方法和callback参数的方法是一样的，也是以json格式来传递。该json字符串就是一个包含所有自定义参数的Key-Value的Map。这里有个特别需要注意的地方就是，用户自定义参数的Key一定要以x:开头，且必须为小写。否则OSS会返回错误。假定用户需要设定两个自定义的参数分别为x:var1和x:var2，对应的值分别为value1和value2，那么构造出来的json格式如下：

{

"x:var1":"value1",

"x:var2":"值2"

}

## 构造回调请求

构造完成上述的callback和callback-var两个参数之后，一共有三种方式传给OSS。其中callback为必填参数，callback-var为可选参数，如果没有自定义参数的话可以不用添加callback-var字段。这三种方式如下：

* 在URL中携带参数。
* 在Header中携带参数。
* 在POST请求的body中使用表单域来携带参数，在使用POST请求上传Object的时候只能使用这种方式来指定回调参数。

这三种方式只能同时使用其中一种，否则OSS会返回InvalidArgument错误。

要将参数附加到OSS的请求中，首先要将上文构造的json字符串使用base64编码，然后按照如下的方法附加到 OSS的请求中：

* 如果在URL中携带参数。把callback=[CallBack]或者callback-var=[CallBackVar]作为一个url参数带入请求发送。计算签名CanonicalizedResource时 ，将callback或者callback-var当做一个subresource计算在内
* 如果在Header中携带参数。把x-oss-callback=[CallBack]或者x-oss-callback-var=[CallBackVar]作为一个head带入请求发送。在计算签名CanonicalizedOSSHeaders时，将x-oss-callback-var和xoss-callback计算在内。如下示例：

|  |
| --- |
| PUT /test.txt HTTP/1.1  Host: callback-test.oss-test.aliyun-inc.com  Accept-ncoding: identity Content-Length: 5  x-oss-callback-var: eyJ4Om15X3ZhciI6ImZvci1jYWxsYmFjay10ZXN0In0=  User-Agent: aliyun-sdk-python/0.4.0 (Linux/2.6.32-220.23.2.ali1089.el5.x86\_64/x86\_64;2.5.4) x-oss-callback:  eyJjYWxsYmFja1VybCI6IjEyMS40My4xMTMuODoyMzQ1Ni9pbmRleC5odG1sIiwgICJjYWxsYmFja0JvZHkiOiJidWNrZX Q9JHtidWNrZXR9Jm9iamVjdD0ke29iamVjdH0mZXRhZz0ke2V0YWd9JnNpemU9JHtzaXplfSZtaW1lVHlwZT0ke21pb WVUeXBlfSZpbWFnZUluZm8uaGVpZ2h0PSR7aW1hZ2VJbmZvLmhlaWdodH0maW1hZ2VJbmZvLndpZHRoPSR7aW1 hZ2VJbmZvLndpZHRofSZpbWFnZUluZm8uZm9ybWF0PSR7aW1hZ2VJbmZvLmZvcm1hdH0mbXlfdmFyPSR7eDpteV  92YXJ9In0=  Host: callback-test.oss-test.aliyun-inc.com  Expect: 100-Continue  Date: Mon, 14 Sep 2015 12:37:27 GMT  Content-Type: text/plain  Authorization: OSS mlepou3zr4u7b14:5a74vhd4UXpmyuudV14Kaen5cY4=  Test |

* 如果需要在POST上传Object的时候附带回调参数会稍微复杂一点，callback参数要使用独立的表单域来附加，如下面的示例：

--9431149156168

Content-Disposition: form-data; name="callback"

eyJjYWxsYmFja1VybCI6IjEwLjEwMS4xNjYuMzA6ODA4My9jYWxsYmFjay5waHAiLCJjYWxsYmFja0hvc3QiOiIxMC4xMD EuMTY2LjMwIiwiY2FsbGJhY2tCb2R5IjoiZmlsZW5hbWU9JChmaWxlbmFtZSkmdGFibGU9JHt4OnRhYmxlfSIsImNhbGx iYWNrQm9keVR5cGUiOiJhcHBsaWNhdGlvbi94LXd3dy1mb3JtLXVybGVuY29kZWQifQ==

如果拥有自定义参数的话，不能直接将callback-var参数直接附加到表单域中，每个自定义的参数都需要使用独立的表单域来附加，举个例子，如果用户的自定义参数的json为

{

"x:var1":"value1",

"x:var2":"value2"

}

那么POST请求的表单域应该如下：

|  |
| --- |
| --9431149156168  Content-Disposition: form-data; name="callback"  eyJjYWxsYmFja1VybCI6IjEwLjEwMS4xNjYuMzA6ODA4My9jYWxsYmFjay5waHAiLCJjYWxsYmFja0hvc3QiOiIxMC4xMD EuMTY2LjMwIiwiY2FsbGJhY2tCb2R5IjoiZmlsZW5hbWU9JChmaWxlbmFtZSkmdGFibGU9JHt4OnRhYmxlfSIsImNhbGx iYWNrQm9keVR5cGUiOiJhcHBsaWNhdGlvbi94LXd3dy1mb3JtLXVybGVuY29kZWQifQ==  --9431149156168  Content-Disposition: form-data; name="x:var1" value1  --9431149156168  Content-Disposition: form-data; name="x:var2" value2 |

同时可以在policy中添加callback条件（如果不添加callback，则不对该参数做上传验证）如：

|  |
| --- |
| { "expiration": "2014-12-01T12:00:00.000Z",  "conditions": [  {"bucket": "johnsmith" }, {"callback":  "eyJjYWxsYmFja1VybCI6IjEwLjEwMS4xNjYuMzA6ODA4My9jYWxsYmFjay5waHAiLCJjYWxsYmFja0hvc3QiOiIxMC4xM  DEuMTY2LjMwIiwiY2FsbGJhY2tCb2R5IjoiZmlsZW5hbWU9JChmaWxlbmFtZSkiLCJjYWxsYmFja0JvZHlUeXBlIjoiYXBwb GljYXRpb24veC13d3ctZm9ybS11cmxlbmNvZGVkIn0="},  ["starts-with", "$key", "user/eric/"],  ]  } |

## 发起回调请求

如果文件上传成功，OSS会根据用户的请求中的callback参数和自定义参数（callback-var参数），将特定内容

以POST方式发送给应用服务器。

POST /index.html HTTP/1.0

Host: 121.43.113.8

Connection: close

Content-Length: 181

Content-Type: application/x-www-form-urlencoded

User-Agent: ehttp-client/0.0.1

bucket=callback-

test&object=test.txt&etag=D8E8FCA2DC0F896FD7CB4CB0031BA249&size=5&mimeType=text%2Fplain&imageInf o.height=&imageInfo.width=&imageInfo.format=&x:var1=for-callback-test

## 返回回调结果

比如应用服务器端返回的回应请求为：

HTTP/1.0 200 OK

Server: BaseHTTP/0.3 Python/2.7.6

Date: Mon, 14 Sep 2015 12:37:27 GMT

Content-Type: application/json

Content-Length: 9

{"a":"b"}

## 返回上传结果

再给客户端的内容为：

HTTP/1.1 200 OK

Date: Mon, 14 Sep 2015 12:37:27 GMT

Content-Type: application/json

Content-Length: 9

Connection: keep-alive

ETag: "D8E8FCA2DC0F896FD7CB4CB0031BA249"

Server: AliyunOSS

x-oss-bucket-version: 1442231779 x-oss-request-id: 55F6BF87207FB30F2640C548

{"a":"b"}

需要注意的是，如果类似CompleteMultipartUpload这样的请求，在返回请求本身body中存在内容（如 XMl格式的信息），使用上传回调功能后会覆盖原有的body的内容如{"a":"b"}，希望对此处做好判断处理。

## 回调签名

用户设置callback参数后，OSS将按照用户设置的callbackUrl发送POST回调请求给用户的应用服务器。应用服务器收到回调请求之后，如果希望验证回调请求确实是由OSS发起的话，那么可以通过在回调中带上签名来验证OSS的身份。

### 生成签名

签名在OSS端发生，采用RSA非对称方式签名，私钥加密的过程为：

authorization = base64\_encode(rsa\_sign(private\_key, url\_decode(path) + query\_string + ‘\n’ + body, md5))

说明：其中private\_key为私钥，只有oss知晓，path为回调请求的资源路径，query\_string为查询字符串

，body为回调的消息体，所以签名过程由以下几步组成：

* 获取待签名字符串：资源路径经过url解码后，加上原始的查询字符串，加上一个回车符，加上回调消息体
* RSA签名：使用秘钥对待签名字符串进行签名，签名的hash函数为md5
* 将签名后的结果做base64编码，得到最终的签名，签名放在回调请求的authorization头中

如下例：

|  |
| --- |
| POST /index.php?id=1&index=2 HTTP/1.0  Host: 121.43.113.8  Connection: close Content-Length: 18 authorization:  kKQeGTRccDKyHB3H9vF+xYMSrmhMZjzzl2/kdD1ktNVgbWEfYTQG0G2SU/RaHBovRCE8OkQDjC3uG33esH2txA== Content-Type: application/x-www-form-urlencoded  User-Agent: ehttp-client/0.0.1 x-oss-pub-key-url: aHR0cDovL2dvc3NwdWJsaWMuYWxpY2RuLmNvbS9jYWxsYmFja19wdWJfa2V5X3YxLnBlbQ== bucket=yonghu-test |

path为/index.php，query\_string为?id=1&index=2，body为bucket=yonghu-test，最终签名结果为 kKQeGTRccDKyHB3H9vF+xYMSrmhMZjzzl2/kdD1ktNVgbWEfYTQG0G2SU/RaHBovRCE8OkQDjC3uG3

3esH2txA==

### 验证签名

验证签名的过程即为签名的逆过程，由应用服务器验证，过程如下：

Result = rsa\_verify(public\_key, md5(url\_decode(path) + query\_string + ‘\n’ + body), base64\_decode(authorization))

字段的含义与签名过程中描述相同，其中public\_key为公钥， authorization为回调头中的签名，整个验证签名

的过程分为以下几步：

* 回调请求的x-oss-pub-key-url头保存的是公钥的url地址的base64编码，因此需要对其做base64解码后获取到公钥，即

public\_key = urlopen(base64\_decode(x-oss-pub-key-url头的值))

这里需要注意，用户需要校验x-oss-pub-key-url头的值必须以http://gosspublic.alicdn.com/或者 https://gosspublic.alicdn.com/开头，目的是为了保证这个publickey是由OSS颁发的。

* 获取base64解码后的签名

signature = base64\_decode(authorization头的值)

* 获取待签名字符串，方法与签名一致

sign\_str = url\_decode(path) + query\_string + ‘\n’ + body

* 验证签名

result = rsa\_verify(public\_key, md5(sign\_str), signature)

以上例为例：

* 获取到公钥的url地址，即

aHR0cDovL2dvc3NwdWJsaWMuYWxpY2RuLmNvbS9jYWxsYmFja19wdWJfa2V5X3YxLnBlbQ= =经过base64解码后得到http://gosspublic.alicdn.com/callback\_pub\_key\_v1.pem

* 签名头

kKQeGTRccDKyHB3H9vF+xYMSrmhMZjzzl2/kdD1ktNVgbWEfYTQG0G2SU/RaHBovRCE8OkQ DjC3uG33esH2txA==做base64解码（由于为非打印字符，无法显示出解码后的结果）

* 获取待签名字符串，即url\_decode(“index.php”) + “?id=1&index=2” + “\n” +

“bucket=yonghu-test”，并做md5

* 验证签名

### 签名示例程序

以下为一段python示例，演示了一个简单的应用服务器，主要是说明验证签名的方法，此示例需要安装

M2Crypto库

|  |
| --- |
| import httplib import base64  import md5  import urllib2  from BaseHTTPServer import BaseHTTPRequestHandler, HTTPServer from M2Crypto import RSA from M2Crypto import BIO  def get\_local\_ip(): try:  csock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM) csock.connect(('8.8.8.8', 80)) (addr, port) = csock.getsockname()  csock.close() return addr except socket.error: return "" class MyHTTPRequestHandler(BaseHTTPRequestHandler): |

|  |
| --- |
| '''  def log\_message(self, format, \*args):  return  '''  def do\_POST(self): #get public key pub\_key\_url = '' try:  pub\_key\_url\_base64 = self.headers['x-oss-pub-key-url'] pub\_key\_url = pub\_key\_url\_base64.decode('base64') if not pub\_key\_url.startswith("http://gosspublic.alicdn.com/") and not pub\_key\_url.startswith("https://gosspublic.alicdn.com/"): self.send\_response(400) self.end\_headers() return  url\_reader = urllib2.urlopen(pub\_key\_url)  #you can cache it pub\_key = url\_reader.read() except:  print 'pub\_key\_url : ' + pub\_key\_url print 'Get pub key failed!' self.send\_response(400) self.end\_headers() return  #get authorization authorization\_base64 = self.headers['authorization'] authorization = authorization\_base64.decode('base64')  #get callback body content\_length = self.headers['content-length'] callback\_body = self.rfile.read(int(content\_length))  #compose authorization string  auth\_str = ''  pos = self.path.find('?') if -1 == pos:  auth\_str = urllib2.unquote(self.path) + '\n' + callback\_body else: auth\_str = urllib2.unquote(self.path[0:pos]) + self.path[pos:] + '\n' + callback\_body print auth\_str  #verify authorization auth\_md5 = md5.new(auth\_str).digest() bio = BIO.MemoryBuffer(pub\_key) rsa\_pub = RSA.load\_pub\_key\_bio(bio) try:  result = rsa\_pub.verify(auth\_md5, authorization, 'md5') except: result = False  if not result:  print 'Authorization verify failed!' print 'Public key : %s' % (pub\_key) |
| print 'Auth string : %s' % (auth\_str)  self.send\_response(400) self.end\_headers() return  #do something accoding to callback\_body  #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)  class MyHTTPServer(HTTPServer): def \_\_init\_\_(self, host, port):  HTTPServer.\_\_init\_\_(self, (host, port), MyHTTPRequestHandler)  if '\_\_main\_\_' == \_\_name\_\_: server\_ip = get\_local\_ip() server\_port = 23451  server = MyHTTPServer(server\_ip, server\_port) server.serve\_forever() |

## 特别须知

如果传入的callback或者callback-var不合法，则会返回400错误，错误码为

”InvalidArgument”，不合法的情况包括以下几类：

* PutObject()和CompleteMultipartUpload()接口中url和header同时传入callback(x-osscallback)或者callback-var(x-oss-callback-var)参数
* callback或者callback-var(PostObject()由于没有callback-var参数，因此没有此限制，下

同)参数过长（超过5KB）

* callback或者callback-var没有经过base64编码
* callback或者callback-var经过base64解码后不是合法的json格式
* callback参数解析后callbackUrl字段包含的url超过限制（5个），或者url中传入的port不合法，比如{"callbackUrl":"10.101.166.30:test", "callbackBody":"test"}
* callback参数解析后callbackBody字段为空
* callback参数解析后callbackBodyType字段的值不是”application/x-www-formurlencoded”或者”application/json”
* callback参数解析后callbackBody字段中变量的格式不合法，合法的格式为${var}
* callback-var参数解析后不是预期的json格式，预期的格式应该为

{"x:var1":"value1","x:var2":"value2"...}

如果回调失败，则返回203，错误码为”CallbackFailed”，回调失败只是表示OSS没有收到预期的回调响应，不代表应用服务器没有收到回调请求（比如应用服务器返回的内容不是json格式），另外，此时文件已经成功上传到了OSS

应用服务器返回OSS的响应必须带有Content-Length的Header，Body大小不要超过1MB。