目录

请求结构

请求结构

客户调用金山云**标签(Tag)**服务的openAPI接口是通过向指定服务地址发送请求，并按照openAPI文档说明在请求中添加相应的公共参数和接口参数来完成的。

标签(Tag)服务openAPI的请求结构组成如下：

1. 服务地址

   标签(Tag)服务接入地址为：tag.region-name.api.ksyun.com

   当前支持的region列表如下

区域	region code
华北1（北京）	cn-beijing-6
华东1（上海）	cn-shanghai-2

2. 通信协议

   支持通过 HTTP 或 HTTPS 两种方式进行请求通信，推荐使用安全性更高的 HTTPS方式发送请求。

3. 请求方法

   标签(Tag)服务的openAPI同时支持GET和POST请求，推荐使用**GET**请求方式。

注意

- 不能混合使用两种请求方式。如果使用 GET 方式，参数均从 querystring 取得；如果使用 POST 方式，参数均从请求Body中取得
- 如果请求方式是GET，需要对所有请求参数做URL编码；如果请求方式是POST，需要使用x-www-form-urlencoded方式进行编码。

4. 请求参数

   金山云openAPI请求包含两类参数：**公共请求参数**和**接口请求参数**。其中，公共请求参数是每个接口都要用到的请求参数，具体可参见公共参数小节；接口请求参数是各个接口所特有的，具体见各个接口的“请求参数”描述。

5. 字符编码

   请求及返回结果都使用UTF-8字符集进行编码。

公共参数

参数说明

公共请求参数是每个标签(Tag)服务都需要使用到的请求参数。

名称	类型	是否必须参数	长度限制(字符)	参数格式	描述
Action	String	是	不确定	[a-zA-Z]+	操作接口名，与调用的具体openAPI相关
Version	String	是	10字符	YYYY-MM-DD	接口版本号，版本号不同接口支持的参数和返回值可能不同，标签（Tag）服务接口当前只支持一个版本，即2016-03-04
X-Amz-Algorithm	String	是	16字符	AWS4-HMAC-SHA256	签名算法，目前只支持一种，即HMAC-SHA256
X-Amz-Credential	String	是	不确定	AccessKeyId/YYYYMMDD/region/service/AWS4_request	信任状信息，包括访问密钥ID，日期，region名称和服务名称以及结尾字符串AWS4_request
X-Amz-Date	String	否（用于覆盖信任状或者date header中的日期）	16字符	ISO 8601 基本格式 YYYYMMDD'T'HHMMSS'Z'，如20151101T120000Z	签名日期
X-Amz-Signature	String	是	64字符	**16进制编码**表示	请求签名值

X-Amz-SignedHeaders	String	是	不确定	[a-zA-Z0-9-;]+	需要在签名计算中包含的请求header
DryRun	Boolean	否	最长5字符	true(1) or false(0)	检查当前调用者是否有权限执行相关操作，而不是真的调用执行相关操作

示例

```
https://tag.cn-shanghai-2.api.ksyun.com?Action=DescribeTags&Version=2016-03-04
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKLTKofy3r07QsilnD_ipmGAIw%2F20161008%2Fcn-shanghai-2%2Ftag%2Faws4_request
&X-Amz-Date=20161008T064016Z
&X-Amz-SignedHeaders=host
&X-Amz-Signature=479586da3ce3eb6ab9d7d3b4d03aeae1ac1634d45a097663b2ff2f0053d04893
&其他接口请求参数
```

返回结果

调用金山云的openAPI服务，调用成功，返回的HTTP状态码（Status）为200；调用失败，返回4xx 或5xx的HTTP状态码（Status）。

金山云的标签（Tag）服务的调用返回的数据格式支持xml和json两种，默认返回xml格式，可通过设置HTTP Header Accept=application/json来改变返回数据格式。

调用成功

- xml格式示例

```
<!--结果的根结点-->
<接口名称+Response>
    <RequestId>4C467B38-3910-447D-87BC-AC049166F216</RequestId>
    <!--返回结果数据-->
</接口名称+Response>
```

- json格式示例

```
{
    "RequestId": "4C467B38-3910-447D-87BC-AC049166F216"
    /*返回结果数据*/
}
```

调用失败

调用接口失败，不会返回结果数据；HTTP请求返回一个4xx或5xx的HTTP状态码，返回的HTTP消息体中包含具体的错误代码(code)及错误信息(message)；与调用成功一样还包含请求ID(RequestId)，在调用方找不到错误原因时，可以联系金山云客服，并提供RequestId，以便我们尽快帮您解决问题。

- xml格式示例

```
<!--结果的根结点-->
<Response> 或者 <ErrorResponse>
    <RequestId>e1eac1b3-1f35-44ba-abd4-7c4b7a9859f3</RequestId>
    <!--返回具体错误消息-->
    <Error>
        <!--错误来源，可省略-->
        <Type>Sender</Type>
        <!--错误代码-->
        <Code>InvalidParameterValue</Code>
        <!--错误消息-->
        <Message>An invalid or out-of-range value was supplied for the input parameter PathPrefix.</Message>
    </Error>
</Response> 或者 <ErrorResponse>
```

- json格式示例

```
{
    "RequestId": "68093a99-2f63-4f39-8f70-3047ab8ecb5b",
    "Error": {
        "Type": "Sender",
        "Code": "InvalidParameterValue",
        "Message": "An invalid or out-of-range value was supplied for the input parameter PathPrefix."
    }
}
```

公共错误

错误代码(Code)	错误消息(Message)	HTTP 状态码	中文描述（语义）
MissingAuthenticationToken	Request is missing 'Host' header.	403	请求header中缺少Host
MissingAuthenticationToken	Request is missing Authentication Token.	403	请求header中缺少认证token
MissingAuthenticationToken	%s not in Http Header.	403	%s不在Http header中
SignatureDoesNotMatch	Host' must be a 'SignedHeader' in the Authorization.	403	请求的SignedHeader中必须包含Host
SignatureDoesNotMatch	Credential should be scoped with a valid terminator: 'aws4_request', not: %s.	403	请求Authorization header中的“Credential”末尾必须是“aws4_request”
SignatureDoesNotMatch	Credential should be scoped to a valid region, not:%s.	403	请求Authorization header中的“Credential”中的Region信息无效
SignatureDoesNotMatch	Credential should be scoped to correct service: %s.	403	请求Authorization header中的“Credential”中的Service信息无效
SignatureDoesNotMatch	The request signature we calculated does not match the signature you provided.	403	请求中提供的签名与实际计算结果不匹配
SignatureDoesNotMatch	Signature expired:%s.	403	签名已过期
SignatureDoesNotMatch	Date in Credential scope does not match YYYYMMDD from ISO-8601 version of date from HTTP.	403	请求Authorization header中的“Credential”中的Date应该是ISO8601基本格式，形如”YYYYMMDD“
InvalidClientTokenId	The security token included in the request is invalid.	403	请求中提供的AccessKeyId无效
AccessDenied	User: %s is not authorized to perform: %s.	403	用户%s无权限操作该资源：%s
IncompleteSignature	Date must be in ISO-8601 'basic format'. Got '%s'. See http://en.wikipedia.org/wiki/ISO_8601.	400	Date必须符合ISO_8601基本格式，参考：http://en.wikipedia.org/wiki/ISO_8601
IncompleteSignature	KSC query-string parameters must include %s. Re-examine the query-string parameters.	400	查询条件中缺少签署信息，查询条件中必须包含”X-Amz-Algorithm“、”X-Amz-Credential“、”X-Amz-SignedHeaders“、”X-Amz-Date“信息
IncompleteSignature	Unsupported ksc 'algorithm': %s.	400	只支持如下签名算法：AWS4-HMAC-SHA256
IncompleteSignature	Authorization header requires 'Credential' parameter. Authorization=%s.	400	请求Authorization header中需要包含“Credential”参数
IncompleteSignature	Credential must have exactly 5 slash-delimited elements, e.g. accesskeyid/date/region/service/aws4_request, got: %s.	400	请求Authorization header中中“Credential”至少包含5项以斜杠分隔的元素，如：keyid/date/region/service/aws4_request
IncompleteSignature	Authorization header format error.	400	请求Authorization header的格式错误
IncompleteSignature	Authorization header requires existence of either a 'X-Amz-Date' or a 'Date' header, Authorization=%s	400	请求中缺少“X-Amz-Date”或者“Date” header信息
IncompleteSignature	Authorization header requires 'Signature' parameter. Authorization=%s	400	请求Authorization header中缺少“Signature”信息
IncompleteSignature	Authorization header requires 'SignedHeaders' parameter. Authorization=%s	400	请求Authorization header中缺少“SignedHeaders”信息
ServiceUnavailable	Exception %s	500	服务暂不可用
ServiceUnavailable	Auth Service is unavailable because of an unknown error, exception or failure	500	验签或授权服务暂不可用

ServiceUnavailable	Request was rejected because it referenced an 'InnerApi' that does not have an internal service	404	请求被拒绝，因其引用的InnerAPI无内部服务。
ServiceUnavailable	OpenAPI or Service is unavailable because of an unknown error, exception or failure.	500	openAPI或服务暂不可用。
DryRunOperation	Request would have succeeded, but DryRun flag is set	412	请求本可成功，但由于设置DryRun标记未成功
NoSuchEntity	Request was rejected because it referenced an 'InnerApi' that does not exist.	404	请求被拒绝，因其引用的InnerAPI不存在
LimitExceeded	Request was rejected because the request speed of this openAPI is beyond the current flow control limit.	409	请求被拒绝，因该openAPI接口访问速度已达到流控上限
InvalidParameterValue	An invalid or out-of-range value was supplied for the input parameter %s.	400	输入参数%s的值无效、不合法或者超出范围
InvalidMethod	The method %s for is not valid for this web service.	400	Method %s对当前web服务无效
MissingParameter	An value must be supplied for the input parameter %s.	400	输入参数 %s的值不能为空
InvalidQueryParameter	The query parameter %s is malformed or does not adhere to KSC standards.	400	查询参数 %s格式不对、不存在或者不符合金山云标准
ServiceTimeout	Internal Service is unavailable because of time out.	500	内部服务由于超时暂不可用

签名机制

标签（Tag）服务的openAPI调用采用AWS签名算法版本4，具体可以参考AWS文档，支持GET和POST两种HTTP方法，GET方法所有请求参数包括signature放置在url中，POST方法则将Signature以名为authorization header的形式放置在header中，其主要区别在于GET方式处理的请求url长度不能过长。

签名计算的主要流程如下：

1. **创建一个正规化请求**

   在签名前，首先将请求进行正规化格式化，目的是让签名计算过程无二意，其主要过程伪代码如下：

   ```
   CanonicalRequest = HTTPRequestMethod + '\n' + CanonicalURI + '\n' + CanonicalQueryString + '\n' + CanonicalHeaders + '\n' + SignedHeaders + '\n' + HexEncode(Hash(RequestPayload))
   ```

   Hash指代计算哈希的算法，目前使用SHA-256，HexEncode是对哈希值进行用16进制编码（使用小写字母）。

   具体步骤如下

   1. 抽取HTTP请求方法（如GET、PUT、POST）结尾附加“换行符”
   2. URI绝对路径进行URI编码得到正规化URI，如果绝对路径为空，那么使用前斜线"/"，结尾附加“换行符”
   3. 构建正规化Querystring，结尾附加“换行符”
      - URI编码每一个querystring参数名称和参数值（注：GET方式需要包含哈希算法、信任状、签名日期和签名header等全部参数）
      - 按照ASCII字节顺序对参数名称严格排序
      - 将排序号的参数名称和参数值用=连接，按照排序结果将“参数对”用&连接
   4. 构建正规化headers，结尾附加“换行符”，伪代码如下：

      ```
      CanonicalHeaders = CanonicalHeadersEntry0 + CanonicalHeadersEntry1 + ... + CanonicalHeadersEntryN
      ```

      其中：

      ```
      CanonicalHeadersEntry = Lowercase(HeaderName) + ':' + Trimall(HeaderValue) + '\n'
      ```

      `lowercase`表示将header名字转为小写字母，`trimall`表示去掉header值前和值后的白空格，并将header值里面的连续白空格变成单空格，但是不去掉双引号中间的任何空格，且最后的正规化headers是按照header名称排序后的结果

   5. 添加签名headers，结尾附加“换行符”。签名header是包含在正规化headers中名称列表，其目的是指明哪些header参与签名计算，从而忽略请求被proxy添加的额外header，其中host、x-amz-date两个header如果存在则必须添加进来，伪代码如下

      ```
      SignedHeaders = Lowercase(HeaderName0) + ';' + Lowercase(HeaderName1) + ";" + ... + Lowercase(HeaderNameN)
      ```

然后处理请求Body，

6. 对请求body使用哈希算法（SHA256）计算哈希值，并将二进制哈希值结果用16进制编码表示出来（且不使用大写字符），伪代码

   ```
   HashedPayload = Lowercase(HexEncode(Hash(requestPayload)))
   ```

   此时，

7. 将上述i-vi步骤的结果连接成一个字符串，即为**正规化请求（Canonical Request）**
8. 将vii步的正规化请求使用vi步的哈希算法计算哈希值

2. **创建签名字符串**

   签名字符串主要包含请求以及正规化请求的元数据信息，由签名算法、请求日期、信任状和正规化请求哈希值连接组成，伪代码如下：

   ```
   StringToSign = Algorithm + '\n' + RequestDate + '\n' + CredentialScope + '\n' + HashedCanonicalRequest
   ```

   其中，签名算法(Algorithm)为AWS4-HMAC-SHA256，请求日期(RequestDate)格式YYYYMMDD'T'HHMMSS'Z' ，信任状(CredentialScope)格式为*YYYYMMDD/region/service*/aws4_request（包括请求日期（ISO 8601 基本格式）），正规化请求哈希值为上述1中第viii步的结果（注意结尾不附加“换行符”）

3. **计算签名信息**

   在计算签名前，首先从私有访问密钥（secret AccessKey）派生出签名密钥（signing key），而不是直接使用私有访问密钥；之后使用签名密钥和2中计算的签名字符串来计算签名值，具体计算过程如下

   1. 生成签名密钥，伪代码如下

      ```
      kSecret = *Your KSC Secret Access Key*
      kDate = HMAC("AWS4" + kSecret, Date)
      kRegion = HMAC(kDate, Region)
      kService = HMAC(kRegion, Service)
      kSigning = HMAC(kService, "aws4_request")
      ```

      其方式是通过HMAC算法依次生成下一个HMAC的key值（第一个为私有访问密钥字符串），而data值则依次为信任状中的各项内容（日期、region、服务、结尾字符串）；HMAC算法采用HMAC-SHA256，返回值为哈希值二进制形式（256bit，32字节），不需要做8/16进制编码显示。

   2. 计算签名,伪代码如下：

      ```
      signature = HexEncode(HMAC(derived-signing-key, string-to-sign))
      ```

      使用HMAC-SHA256算法，以签名密钥作为key，签名字符串作为data计算签名，签名后的二进制哈希值结果以16进制编码输出。