目录

公共参数

参数说明

公共请求参数是每个**金山云OpenAPI接口**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，具体请查看API接口说明文档
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://iam.api.ksyun.com/?
Action=ListUsers&Version=2015-11-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKLTGoOpHK-EQWiDZWTSBSll2Q%2F20160914%2Fcn-beijing-6%2Fiam%2Faws4_request
&X-Amz-Date=20160914T114902Z
&X-Amz-SignedHeaders=host
&X-Amz-Signature=88f6284257863dedfc350da05d19d07f76cca622e93b829f5ce26c1a75d3da39
&接口请求参数
```

签名机制

金山云openAPI调用支持用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：

- 步骤一：创建一个正规化请求。
- 步骤二：将正规化请求进行哈希处理。
- 步骤三：创建签名字符串。
- 步骤四：计算签名。

步骤一： 创建一个正规化请求

在签名前，请按照以下步骤创建正规化请求，确保金山云在收到请求时计算出的签名与您计算出的签名相同，否则请求将被拒绝。 **示例：正规化请求伪代码**

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

在此伪代码中： （1）Hash指代计算哈希的算法，目前使用SHA-256。 （2）HexEncode表示以小写字母形式返回摘要的 base-16 编码的函数。例如，HexEncode("m") 返回值 6d 而不是 6D。输入的每一个字节都必须表示为两个十六进制字符。 （3）参数含义说明 HTTPRequestMethod:HTTP正规化请求方法参数

- CanonicalURI：正规化URI参数
- CanonicalQueryString：正规化Query参数
- CanonicalHeaders：正规化Header参数
- SignedHeaders：正规化签名的Header参数
- HexEncode(Hash(RequestPayload))：正规化body参数

要创建正规化请求，请将以下来自每个步骤的部分连接为一个字符串：

1. 抽取HTTP请求方法（如GET、PUT、POST）结尾附加“换行符”。

2. 添加正规化URI参数，后跟换行符。

- 正规化URI是将绝对路径行URI编码.
- 如果绝对路径为空，那么使用前斜线"/"，结尾附加“换行符”。

3.添加正规化Query参数数，后跟换行符。

如果请求不包括查询字符串，请使用空字符串（实际上是空白行）。

要构建正规化查询字符串，请完成以下步骤： a.按照ASCII字节顺序对参数名称严格排序(具有重复名称的参数应按值进行排序)。 b.对Querystring的每个参数名称和值进行 URI 编码。（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） c.以排序后的列表中第一个参数名称开头，构造正规化查询字符串。 d.对于每个参数，追加 URI 编码的参数名称，后跟等号字符 (=)，再接 URI 编码的参数值。对没有值的参数使用空字符串。 e.在每个参数值后追加与字符(&)，列表中最后一个值除外。

4.添加正规化Header参数，后跟换行符。

正规化Header参数包括您要包含在签名请求中的所有 HTTP Header参数的列表。具体构造规则如下： a.请按照ASCII字节顺序对header名称严格排序。 b.请将所有Header参数名称转换为小写形式。 c使用冒号（:）连接参数名称和参数值。参数值有多个时使用分号（“;”）来分隔。请勿对有多个值的Header进行值排序。 d.添加一个新行（“\n”）。

以下伪代码描述如何构造正规化Header参数列表：

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

其中：

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

- Lowercase 表示将所有字符转换为小写字母的函数。
- Trimall 函数删除值前后的多余空格并将连续空格转换为单个空格，但是不去掉双引号中间的任何空格。

5.添加正规化签名的Header参数，后跟换行符。 该值是您包含在正规化Header参数中的Header列表。通过添加此Header参数列表，您可以向金山云告知请求中的哪些Header参数是签名过程的一部分以及在验证请求时金山云可以忽略哪些Header参数（例如，由代理添加的任何附加Header参数）。

其中如果header里存在host、x-amz-date，则必须添加进来。

具体构造规则如下：

a.请按照ASCII字节顺序对header名称严格排序。 b.请将所有Header参数名称转换为小写形式。 c.使用分号（“;”）来分隔这些Header参数名称。最后一个Header参数无需加分号。

以下伪代码描述如何构造签名Header参数列表。Lowercase 表示将所有字符转换为小写字母的函数。

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

6.添加正规化签名的Header参数。 对请求body使用哈希算法（SHA256）计算哈希值，必须以小写十六进制字符串形式表示。

如果body为空，则使用空字符串作为哈希函数的输入。

伪代码如下：

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

7.将上诉1-6步骤中的结果连接成一个字符串，即为正规化请求。

步骤二：将正规化请求进行哈希处理

将步骤一中得到的正规划请求使用哈希算法（SHA256）计算哈希值，必须以小写十六进制字符串形式表示。

步骤三：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。 **待签字符串结构伪代码：**

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

其中：

- Algorithm：签名算法固定为AWS4-HMAC-SHA256
- RequestDate：请求日期格式为格式YYYYMMDD'T'HHMMSS'Z'
- CredentialScope：信任状格式为 *YYYYMMDD/region/service*/aws4_request（包括请求日期（ISO 8601 基本格式））
- HashedCanonicalRequest：正规化请求哈希值为上述步骤二的结果，注意结果不要附加换行符。

步骤四：计算签名字符串

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）。派生签名密钥会指定日期、服务和区域，可以提供了更高程度的保护。

1.生成派生签名密钥

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")
```

Access Key:可以为主账号或子用户的访问密钥（secret AccessKey） Date:请求的日期格式为YYYYMMDD（例如，20150830），不包括时间。 Region：要访问的目标区域。每个服务支持的region可能不同，详见各服务openapi文档说明。 Service:要访问的目标服务简称，一般可通过服务接入地址获取。接入地址格式一般为{service}.{region}.api.ksyun.com或{service}.api.ksyun.com。如访问控制的服务接入地址为访问控制的服务接入地址为：iam.api.ksyun.com，其中sevice为iam。

其中 HMAC(key, data) 表示以二进制格式返回输出的 HMAC-SHA256 函数。每个哈希函数的结果将成为下一个函数的输入。请注意：

- 哈希过程中所使用的日期的格式为 YYYYMMDD（例如，20150830），不包括时间。
- 确保以正确的顺序为您要使用的编程语言指定 HMAC 参数。在此示例中，密钥是第一个参数，数据（消息）是第二个参数，但您使用的函数可能以不同顺序指定密钥和数据。
- 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。

示例输入：

```
HMAC(HMAC(HMAC(HMAC("AWS4" + kSecret,"20150830"),"cn-beijing-6"),"iam"),"aws4_request")
```

示例派生签名密钥：

```
c4afb1cc5771d871763a393e44b703571b55cc28424d1a5e86da6ed3c154a4b9
```

步骤五：计算签名

请使用派生的签名密钥和待签字符串作为加密哈希函数的输入。在计算签名后，将二进制值转换为十六进制表示形式。

伪代码如下：

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

确保以正确的顺序为您要使用的编程语言指定 HMAC 参数。在此示例中，密钥是第一个参数，数据（消息）是第二个参数，但您使用的函数可能以不同顺序指定密钥和数据。

示例签名：

5d672d79c15b13162d9279b0855cfba6789a8edb4c82c400e06b5924a6f2b5d7

SDK

金山云提供以下语言包的SDK：

- Java SDK：查看详情
- Python SDK：查看详情
- PHP SDK：查看详情
- Go SDK：查看详情