API签名规范

请求样例

```
Authorization: XYXY-HMAC-SHA256 Credential=1FihRrMitxji/20120525/
zh-cn-shanghai/xyxy-service/xyxy_request, SignedHeaders=host;x-xy-date,

Signature=79b5ab08173ad52554d3ffaa60ded9fc832d357835a9eb8748a9e08580bda
b74
```

Authorization HTTP请求Header头标识

XYXY-HMAC-SHA256 XYXY签名算法-特定标识

Credential 认证路径串，组成是ACCESS_KEY/X-Xy-Date HTTP头的值/服务区域/服务名称/XYXY服务请求标识

SignedHeaders 参与签名计算的Header

Signature 签名值

规范请求

```
CanonicalRequest =

HTTPRequestMethod + '\n' +

CanonicalURI + '\n' +

CanonicalQueryString + '\n' +

CanonicalHeaders + '\n' +

SignedHeaders + '\n' +

HexEncode(Hash(RequestPayload))
```

HTTPRequestMethod 约定成大写的Request method，例如 GET POST。

CannonicalURI 规范 URI 是 URI 的绝对路径部分的 URI 编码版本，该版本是 URI 中的一切 - 从 HTTP 主机到开始查询字符串参数（如果有）的问号字符（“?”）。
对请求url（补上http://前缀）encode处理，java写法 new URI("http://" + encoded).normalize().getRawPath()。

CannonicalQueryString 添加规范查询字符串，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）。

要构建规范查询字符串，请完成以下步骤：

- 按字符代码点以升序顺序对参数名称进行排序。例如，以大写字母 F 开头的参数名称排在以小写字母 b 开头的参数名称之前。
- 根据以下规则对每个参数名称和值进行 URI 编码：
- ◦ 请勿对 RFC 3986 定义的任何非预留字符进行 URI 编码，这些字符包括：A-Z、a-z、0-9、连字符 (-)、下划线 (_)、句点 (.) 和波浪符 (~)。
- ◦ 使用 %XY 对所有其他字符进行百分比编码，其中“X”和“Y”为十六进制字符（0-9 和大写字母 A-F）。例如，空格字符必须编码为 %20（不像某些编码方案那样使用“+”），扩展 UTF-8 字符必须采用格式 %XY%ZA%BC。
- 以排序后的列表中第一个参数名称开头，构造规范查询字符串。
- 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。
- 在每个参数值后追加与字符 (&)，列表中最后一个值除外。

CannonicalHeaders 规范标头包括您要包含在签名请求中的所有 HTTP 标头的列表。

- 您必须至少包含 x-ke-date 标头。
- 要创建规范标头列表，请将所有标头名称转换为小写形式并删除前导空格和尾随空格。将标头值中的连续空格转换为单个空格。
- 通过按字符代码对（小写）标头排序，然后对标头名称进行迭代操作，来构建规范标头列表。根据以下规则构造每个标头：
- ◦ 追加小写标头名称，后跟冒号。
- ◦ 追加该标头的值的逗号分隔列表。请勿对有多个值的标头进行值排序。
- ◦ 追加一个新行（“\n”）。

SignedHeaders 要创建已签名标头列表，请将所有标头名称转换为小写形式，按字符代

码对其进行排序，并使用分号来分隔这些标头名称。

RequestPayload 应用SHA-256来对body的值进行hash计算，比如

```
// POST
String contentForPOST = "{\"ke_id\":120933,\"iss\":\"Tao-Yang\"}";
String contentHashForPOST = Sha256.get(contentForPOST, Charset.forName(
"UTF8"));
```

待签字符串结构

```
StringToSign =

Algorithm + \n +

RequestDateTime + \n +

CredentialScope + \n +

HashedCanonicalRequest
```

Algorithm 签名算法 XYXY-HMAC-SHA256

RequestDateTime 追加请求日期值。该日期是使用 ISO8601 基本格式以 YYYYMMDD'T'HHMMSS'Z' 格式在 x-xy-date 标头中指定的。

CredentialScope 追加凭证范围值。此值是一个字符串，包含日期、目标区域、所请求的服务和小写字符形式的终止字符串（“xyxy_request”）。区域和服务名称字符串必须采用 UTF-8 编码。

组成规则 date的 YYYYMMDD/region/service/xyxy_request

```
20120525/zh-cn-shanghai/xyxy-service/xyxy_request
```

HashedCanonicalRequest 应用SHA-256来对上一步**CanonicalRequest**进行hash计算。

样例，待签字符串

```
XYXY-HMAC-SHA256
20150830T123600Z
```

```
20150830/us-east-1/service/xyxy_request
da61028f9d164f47170b70dae4b6c08fab4457bc8c01a58d3778c69a6fe11eb0
```

派生签名秘钥

```
kSecret = your secret access key

kDate = HMAC("XYXY" + kSecret, Date)

kRegion = HMAC(kDate, Region)

kService = HMAC(kRegion, Service)

kSigning = HMAC(kService, "xyxy_request")
```

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。

将签名信息添加到 Authorization 标头

也就是文章开头的部分。

Authorization: XYXY-HMAC-SHA256
Credential=1FihRrMitxji/20120525/zh-cn-shanghai/xyxy-service/xyxy_request,
SignedHeaders=host;x-xy-date,
Signature=79b5ab08173ad52554d3ffaa60ded9fc832d357835a9eb8748a9e08580bdab74

参考

aws v4