Alibaba Cloud API 网关

用户指南（调用 API）

文档版本：20200515

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. 阿里云文档中所有内容，包括但不限于图片、架构设计、页面布局、文字描述，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
	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。	 **禁止：** 重置操作将丢失用户配置数据。
	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。
	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。
	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。	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 客户端签名说明文档

发布的API如果如果使用摘要签名认证方式（APP Key和APP Secret），客户端在调用API时，需要使用签名秘钥对请求内容进行签名计算，并将签名同步传输给服务器端进行签名验证。API网关提供的SDK内置了签名实现，您只需要将签名秘钥配置在SDK中，即可实现发起携带正确签名的请求。如果您需要自己在客户端实现签名计算过程，可以参考本文档。

一.签名与验签原理

API网关提供前端签名及验签能力，前端签名及验签主要两点用途：

- 验证客户端请求的合法性，确认请求中携带授权后的AK生成的签名；
- 防止请求数据在网络传输过程中被篡改。

API的拥有者可以在API网关控制台的应用管理页面生成APP，每个APP会携带一对签名秘钥（APP Key和APP Secret）,API拥有者将API授权给指定的APP（APP可以是API拥有者颁发或者API调用者所有）后，API调用者就可以用APP的签名秘钥来调用相关的API了。

客户端调用 API 时，需要使用已授权签名秘钥对请求内容的关键数据进行加密签名计算，并且将APP Key和加密后生成的字符串放在请求的 Header 传输给API网关，API网关会读取请求中的APP Key的头信息，并且根据APP Key的值查询到对应的APP Secret的值，使用APP Secret对收到的请求中的关键数据进行签名计算，并且使用自己的生成的签名和客户端传上来的签名进行比对，来验证签名的正确性。只有签名验证通过的请求才会发送给后端服务，否则API网关会认为该请求为非法请求，直接返回错误应答。

API网关只会验证安全认证类型为“阿里云APP”或者“OpenID Connect & 阿里云APP”两种类型的API请求的签名，其他安全类型的API请求不会验证签名。

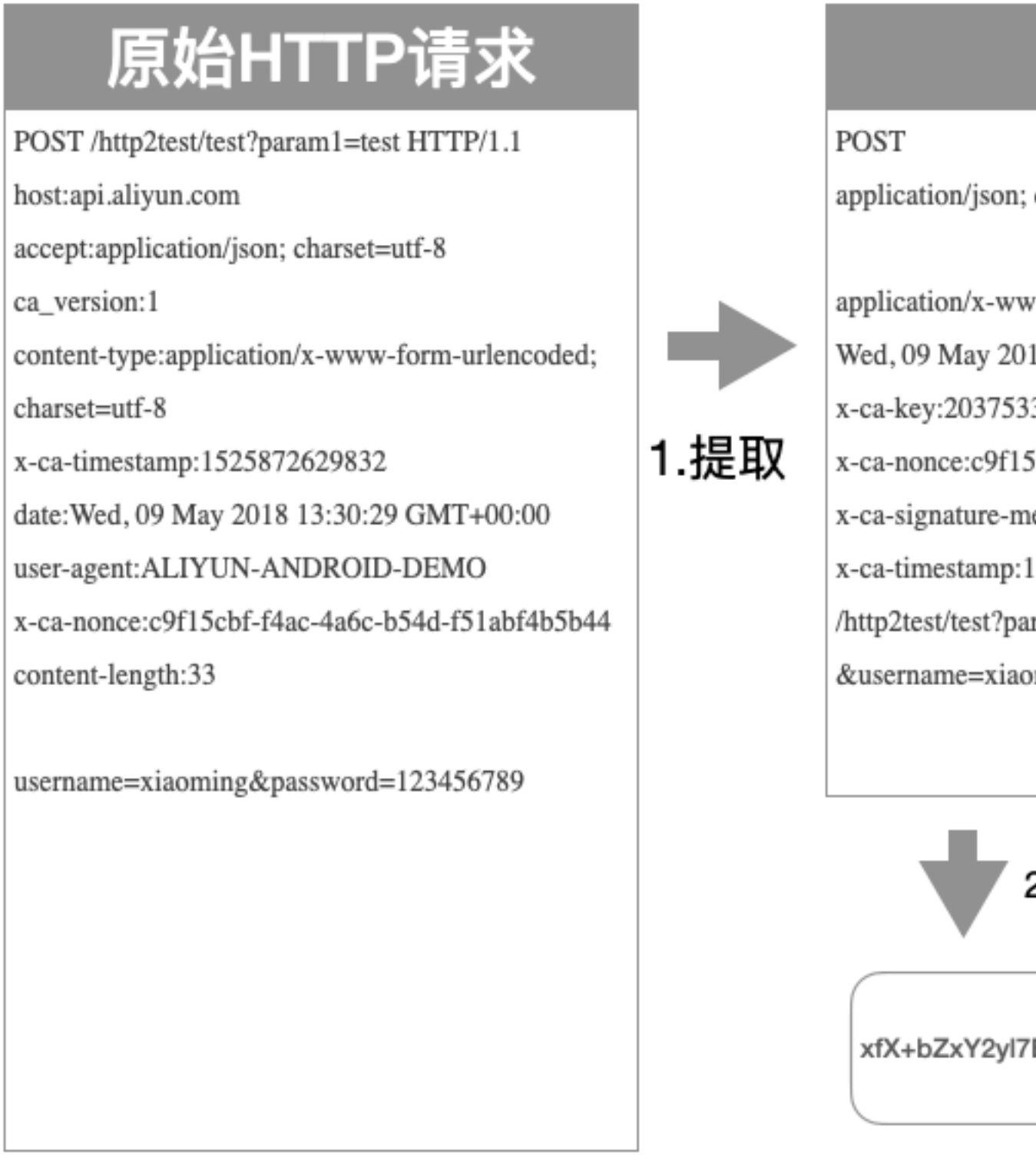
1.1 前置条件

API的调用方需要在调用API之前获取到已经授权给对应API的签名秘钥对：

- APP Key
- APP Secret

被调用的API的安全认证类型为“阿里云APP”或者“OpenID Connect & 阿里云APP”两种类型之一。

1.2 客户端生成签名



客户端生成签名一共分三步处理：

1. 从原始请求中提取关键数据，得到一个用来签名的字符串；
2. 使用加密算法加APP Secret对关键数据签名串进行加密处理，得到签名；
3. 将签名所相关的所有头加入到原始HTTP请求中，得到最终HTTP请求。

1.3 服务器端验证签名

服务器验证客户端签名一共分四步处理：

1. 从接收到的请求中提取关键数据，得到一个用来签名的字符串；
2. 从接收到的请求中读取APP Key，通过APP Key查询到对应的APP Secret；

3. 使用加密算法和APP Secret对关键数据签名串进行加密处理，得到签名；
4. 从接收到的请求中读取客户端签名，对比服务器端签名和客户端签名的一致性。

二.生成与传递签名

2.1 提取签名串

客户端需要从Http请求中提取出关键数据，组合成一个签名串，生成的签名串的格式如下：

```
HTTPMethod
Accept
Content-MD5
Content-Type
Date
Headers
PathAndParameters
```

以上7个字段构成整个签名串，字段之间使用\n间隔，如果Headers为空，则不需要加\n，其他字段如果为空都需要保留\n。签名大小写敏感。下面介绍下每个字段的提取规则：

- HTTPMethod：HTTP的方法，全部大写，比如POST
- Accept：请求中的Accept头的值，可为空。建议显示设置 Accept Header。当 Accept 为空时，部分 Http 客户端会给 Accept 设置默认值为 */*，导致签名校验失败。
- Content-MD5：请求中的Content-MD5头的值，可为空只有在请求存在Body且Body为非Form形式时才计算Content-MD5头，下面是Java的Content-MD5值的参考计算方式：

```
String content-MD5 = Base64.encodeBase64(MD5(bodyStream.getbytes("UTF-8")));
```

- Content-Type：请求中的Content-Type头的值，可为空
- Date：请求中的Date头的值，可为空
- Headers用户可以选取指定的header参与签名，关于header的签名串拼接方式有以下规则：
  - 参与签名计算的Header的Key**按照字典排序**后使用如下方式拼接

```
HeaderKey1 + ":" + HeaderValue1 + "\n"\+
HeaderKey2 + ":" + HeaderValue2 + "\n"\+
...
HeaderKeyN + ":" + HeaderValueN + "\n"
```

  - 某个Header的Value为空，则使用HeaderKey+":"+"\n"参与签名，需要保留Key和英文冒号。
  - 所有参与签名的Header的Key的集合使用英文逗号分割放到Key为X-Ca-Signature-Headers的Header中；
  - 以下Header不参与Header签名计算：X-Ca-Signature、X-Ca-Signature-Headers、Accept、Content-MD5、Content-Type、Date。

- PathAndParameters这个字段包含Path，Query和Form中的所有参数，具体组织形式如下：

```
Path + "?" + Key1 + "=" + Value1 + "&" + Key2 + "=" + Value2 + ... "&" + KeyN + "=" + ValueN
```

  - Query和Form参数对的Key按照字典排序后使用上面的方式拼接；
  - Query和Form参数为空时，则直接使用Path，不需要添加？；
  - 参数的Value为空时只保留Key参与签名，等号不需要再加入签名；
  - Query和Form存在数组参数时（key相同，value不同的参数），取第一个Value参与签名计算。

下面我们看一个例子，这里有一个普通的HTTP请求：

```
POST /http2test/test?param1=test HTTP/1.1
host:api.aliyun.com
accept:application/json; charset=utf-8
ca_version:1
content-type:application/x-www-form-urlencoded; charset=utf-8
x-ca-timestamp:1525872629832
date:Wed, 09 May 2018 13:30:29 GMT+00:00
user-agent:ALIYUN-ANDROID-DEMO
x-ca-nonce:c9f15cbf-f4ac-4a6c-b54d-f51abf4b5b44
content-length:33

username=xiaoming&password=123456789
```

生成的正确签名串为：

```
POST
application/json; charset=utf-8

application/x-www-form-urlencoded; charset=utf-8
Wed, 09 May 2018 13:30:29 GMT+00:00
x-ca-key:203753385
x-ca-nonce:c9f15cbf-f4ac-4a6c-b54d-f51abf4b5b44
x-ca-signature-method:HmacSHA256
x-ca-timestamp:1525872629832
/http2test/test?param1=test&password=123456789&username=xiaoming
```

2.2 计算签名

客户端从HTTP请求中提取出关键数据组装成签名串后，需要对签名串进行加密及编码处理，形成最终的签名，目前API网关支持以下两种加密算法：

- HmacSHA256
- HmacSHA1

具体的加密形式如下，其中stringToSign是提取出来的签名串，secret是APP Secret：

```
Mac hmacSha256 = Mac.getInstance("HmacSHA256");
```

```
hmacSha256.init(new SecretKeySpec(secret.getBytes("UTF-8"), 0, keyBytes.length, "
HmacSHA256"));
byte[] md5Result = hmacSha1.doFinal(stringToSign.getBytes("UTF-8"));
String sign = Base64.encodeBase64String(md5Result);
```

```
Mac hmacSha1 = Mac.getInstance("HmacSHA1");
hmacSha1.init(new SecretKeySpec(secret.getBytes("UTF-8"), 0, keyBytes.length, "
HmacSHA1"));
byte[] md5Result = hmacSha1.doFinal(stringToSign.getBytes("UTF-8"));
String sign = Base64.encodeBase64String(md5Result);
```

抽象一下就是将串（StringToSign）使用UTF-8解码后得到Byte数组，然后使用加密算法对Byte数组进行加密，然后使用Base64算法进行编码，形成最终的签名。

2.3 传输签名

客户端需要将以下四个Header放在HTTP请求中传输给API网关，进行签名校验：

- x-ca-key：取值APP Key，必选；
- x-ca-signature-method：签名算法，取值HmacSHA256或者HmacSHA1，可选，默认值为HmacSHA256；
- X-Ca-Signature-Headers：所有签名头的Key的集合，使用英文逗号分隔，可选；
- X-Ca-Signature：签名，必选；

下面是携带签名的整个HTTP请求的示例：

```
POST /http2test/test?param1=test HTTP/1.1
host:api.aliyun.com
accept:application/json; charset=utf-8
ca_version:1
content-type:application/x-www-form-urlencoded; charset=utf-8
x-ca-timestamp:1525872629832
date:Wed, 09 May 2018 13:30:29 GMT+00:00
user-agent:ALIYUN-ANDROID-DEMO
x-ca-nonce:c9f15cbf-f4ac-4a6c-b54d-f51abf4b5b44
x-ca-key:203753385
x-ca-signature-method:HmacSHA256
x-ca-signature-headers:x-ca-timestamp,x-ca-key,x-ca-nonce,x-ca-signature-method
x-ca-signature:xfX+bZxY2yl7EB/qdoDy9v/uscw3Nnj1pgoU+Bm6xdM=
content-length:33

username=xiaoming&password=123456789
```

三.签名排错方法

API网关签名校验失败时，会将服务端的签名串（StringToSign）放到HTTP Response的Header中返回到客户端，Key为：X-Ca-Error-Message，用户只需要将本地计算的签名串（StringToSign）与服务端返回的签名串进行对比即可找到问题；

如果服务端与客户端的StringToSign一致请检查用于签名计算的APP Secret是否正确；

因为HTTP Header中无法表示换行，因此StringToSign中的换行符都被替换成#。

```
errorMessage：Invalid Signature, Server StringToSign:`GET#application/json##applicatio
n/json##X-Ca-Key:200000#X-Ca-Timestamp:1589458000000#/app/v1/config/keys?keys
=TEST`
```

说明服务器的签名是：

```
GET
application/json

application/json

X-Ca-Key:200000
X-Ca-Timestamp:1589458000000
/app/v1/config/keys?keys=TEST
```

四.签名Java实现示例

签名计算的详细实现可以参考API网关提供的SDK，在API网关控制台的下面两个页面中可以下载Java/Android/Objective-C的带源码的SDK：

- 开放API-SDK/文档自动生成
- 调用API-已授权的API的SDK

Java/Andoird的签名实现具体参考类：com.alibaba.cloudapi.sdk.util.SignUtil