

标签 API 文档



腾讯云

【版权声明】

©2013–2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】



及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。
您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

API 文档

更新历史

简介

API 概览

调用方式

请求结构

公共参数

签名方法 v3

签名方法

返回结果

参数类型

写接口

批量创建标签

批量删除标签

为资源解绑标签

为资源绑定标签

读接口

查询资源标签列表

查询标签键列表

查询标签值列表

获取标签列表

项目相关接口

获取项目列表

创建项目

修改项目

数据结构

错误码

标签 API 2017

API 概览

调用方式

请求结构

返回结果

公共参数

请求签名

写接口

创建标签

修改资源标签值

批量操作资源所有标签

标签关联资源

资源标签解绑

删除标签

读接口

查询标签列表

查询标签键

查询标签值

查看资源关联的标签

错误码

API 文档

更新历史

最近更新时间：2024-01-26 01:20:14

第 14 次发布

发布时间：2024-01-26 01:20:11

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- [DescribeProjects](#)
 - 新增入参：ProjectId, ProjectName

第 13 次发布

发布时间：2023-09-21 05:27:30

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- [DescribeResourceTagsByResourceIds](#)
 - 新增入参：Category
- [DescribeTagKeys](#)
 - 新增入参：Category
- [DescribeTagValues](#)
 - 新增入参：Category
- [GetTagKeys](#)
 - 新增入参：Category
- [GetTagValues](#)
 - 新增入参：Category
- [GetTags](#)
 - 新增入参：Category

修改数据结构：

- [Tag](#)
 - 新增成员：Category
- [TagResource](#)
 - 新增成员：Category
- [TagWithDelete](#)
 - 新增成员：Category

第 12 次发布

发布时间：2022-10-26 06:58:19

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [AddProject](#)
- [UpdateProject](#)

第 11 次发布

发布时间：2022-08-08 06:36:01

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [DescribeProjects](#)

新增数据结构：

- [Project](#)

第 10 次发布

发布时间：2022-04-08 06:12:35

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [CreateTags](#)
- [DeleteTags](#)
- [GetResources](#)
- [GetTagKeys](#)
- [GetTagValues](#)
- [GetTags](#)
- [TagResources](#)
- [UnTagResources](#)

新增数据结构：

- [FailedResource](#)
- [ResourceTagMapping](#)

第 9 次发布

发布时间：2020-07-30 08:15:41

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [DescribeResourcesByTagsUnion](#)

第 8 次发布

发布时间：2020-07-14 08:17:00

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [AttachResourcesTag](#)
- [DescribeResourceTagsByResourceIdsSeq](#)
- [DescribeTagValuesSeq](#)
- [DescribeTagsSeq](#)
- [DetachResourcesTag](#)
- [ModifyResourcesTagValue](#)

第 7 次发布

发布时间：2020-04-07 08:13:55

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- [DescribeTags](#)
 - 新增入参：ShowProject

第 6 次发布

发布时间：2020-04-02 08:15:07

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [DescribeResourceTagsByTagKeys](#)

修改接口：

- [DescribeTagKeys](#)
 - 新增入参：ShowProject
- [DescribeTags](#)
 - 新增入参：TagKeys

新增数据结构：

- [ResourceIdTag](#)

第 5 次发布

发布时间：2020-04-01 08:37:05

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [DescribeResourceTags](#)

第 4 次发布

发布时间：2020-03-26 08:12:32

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- [TagResource](#)
 - 新增成员：ServiceType

第 3 次发布

发布时间：2019-10-11 21:04:23

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- [DescribeResourcesByTags](#)
 - 修改入参：TagFilters
 - 修改出参：Rows

新增数据结构：

- [ResourceTag](#)
- [TagFilter](#)

删除数据结构：

- ResourcesTag
- TagFilters

第 2 次发布

发布时间：2019-08-22 20:11:03

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [DescribeResourcesByTags](#)

新增数据结构：

- ResourcesTag
- TagFilters

第 1 次发布

发布时间：2019-06-06 20:32:16

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [AddResourceTag](#)
- [CreateTag](#)
- [DeleteResourceTag](#)
- [DeleteTag](#)
- [DescribeResourceTagsByResourceIds](#)
- [DescribeTagKeys](#)
- [DescribeTagValues](#)
- [DescribeTags](#)
- [ModifyResourceTags](#)
- [UpdateResourceTagValue](#)

新增数据结构：

- [Tag](#)
- [TagKeyObject](#)
- [TagResource](#)
- [TagWithDelete](#)

简介

最近更新时间：2025-04-25 01:54:32

随着腾讯云用户资源数量的增加，用户管理资源的难度也随之增加。为方便用户更快速有效地查询和管理各种资源，腾讯云推出标签这一产品。人们可以通过标签来进行腾讯云已有资源的分类管理，通过预设标签来进行资源规划。标签是一些充当元数据的词和短语，用于标识和组织腾讯云资源。标签限制随资源而有所不同，但大多数最多可以有 50 个标签。

注意：

- 本章节访问管理 API 接口均为最新标签 3.0 接口，后续标签相关新增功能都会在此章节更新。我们强烈推荐您使用 最新 API3.0 接口。
- 旧版 API 接口功能将保持不变，未来可能停止维护，如您仍需查看旧版接口可参考：[标签 API（旧版）概览](#)。但强烈建议您及时切换为 最新 API3.0 接口。

API 概览

最近更新时间：2024-08-02 02:05:33

写接口

接口名称	接口功能	频率限制（次/秒）
AddResourceTag	标签关联资源	20
AttachResourcesTag	标签关联批量资源	20
CreateTag	创建标签	20
CreateTags	批量创建标签	20
DeleteResourceTag	标签解绑资源	20
DeleteTag	删除标签	20
DeleteTags	批量删除标签	20
DetachResourcesTag	标签解绑批量资源	20
ModifyResourceTags	批量修改资源关联的标签	20
ModifyResourcesTagValue	修改批量资源关联的标签值	20
UnTagResources	为资源解绑标签	20
UpdateResourceTagValue	修改资源标签值	20
TagResources	为资源绑定标签	20

读接口

接口名称	接口功能	频率限制（次/秒）
DescribeResourceTags	查询资源关联标签	20
DescribeResourceTagsByResourceIds	批量查看资源关联的标签	100
DescribeResourceTagsByResourceIdsSeq	按顺序查看资源关联的标签	20
DescribeResourceTagsByTagKeys	根据标签键获取资源标签	20
DescribeResourcesByTags	通过标签查询资源列表	100
DescribeResourcesByTagsUnion	通过标签查询资源列表并集	20
DescribeTagKeys	查询标签键	20
DescribeTagValues	查询标签值	20
DescribeTagValuesSeq	按顺序查询标签值	20
DescribeTags	查询标签列表	60
DescribeTagsSeq	按顺序查询标签列表	20
GetResources	查询资源标签列表	20

接口名称	接口功能	频率限制（次/秒）
GetTagKeys	查询标签键列表	20
GetTagValues	查询标签值列表	20
GetTags	获取标签列表	20

项目相关接口

接口名称	接口功能	频率限制（次/秒）
DescribeProjects	获取项目列表	20
AddProject	创建项目	20
UpdateProject	修改项目	20

⚠ 注意：

以上给出的接口频率限制维度为 API + 接入地域 + 子账号，有关限频更多说明参考：[API 频率限制说明](#)

调用方式

请求结构

最近更新时间：2025-05-21 01:29:45

1. 服务地址

API 支持就近地域接入，本产品就近地域接入域名为 tag.tencentcloudapi.com，也支持指定地域域名访问，例如广州地域的域名为 tag.ap-guangzhou.tencentcloudapi.com。

推荐使用就近地域接入域名。根据调用接口时客户端所在位置，会自动解析到最近的某个具体地域的服务器。例如在广州发起请求，会自动解析到广州的服务器，效果和指定 tag.ap-guangzhou.tencentcloudapi.com 是一致的。

注意：对时延敏感的业务，建议指定带地域的域名。

注意：域名是 API 的接入点，并不代表产品或者接口实际提供服务的地域。产品支持的地域列表请在调用方式/公共参数文档中查阅，接口支持的地域请在接口文档输入参数中查阅。

目前支持的域名列表为：

接入地域	域名
就近地域接入（推荐，只支持非金融区）	tag.tencentcloudapi.com
华南地区（广州）	tag.ap-guangzhou.tencentcloudapi.com
华东地区（上海）	tag.ap-shanghai.tencentcloudapi.com
华东地区（南京）	tag.ap-nanjing.tencentcloudapi.com
华北地区（北京）	tag.ap-beijing.tencentcloudapi.com
西南地区（成都）	tag.ap-chengdu.tencentcloudapi.com
西南地区（重庆）	tag.ap-chongqing.tencentcloudapi.com
港澳台地区（中国香港）	tag.ap-hongkong.tencentcloudapi.com
亚太东南（新加坡）	tag.ap-singapore.tencentcloudapi.com
亚太东南（雅加达）	tag.ap-jakarta.tencentcloudapi.com
亚太东南（曼谷）	tag.ap-bangkok.tencentcloudapi.com
亚太东北（首尔）	tag.ap-seoul.tencentcloudapi.com
亚太东北（东京）	tag.ap-tokyo.tencentcloudapi.com
美国东部（弗吉尼亚）	tag.na-ashburn.tencentcloudapi.com
美国西部（硅谷）	tag.na-siliconvalley.tencentcloudapi.com
南美地区（圣保罗）	tag.sa-saopaulo.tencentcloudapi.com
欧洲地区（法兰克福）	tag.eu-frankfurt.tencentcloudapi.com

注意：由于金融区和非金融区是隔离不互通的，因此当访问金融区服务时（公共参数 Region 为金融区地域），需要同时指定带金融区地域的域名，最好和 Region 的地域保持一致。

金融区接入地域	金融区域名
华东地区（上海金融）	tag.ap-shanghai-fsi.tencentcloudapi.com
华南地区（深圳金融）	tag.ap-shenzhen-fsi.tencentcloudapi.com

2. 通信协议

腾讯云 API 的所有接口均通过 HTTPS 进行通信，提供高安全性的通信通道。

3. 请求方法

支持的 HTTP 请求方法：

- POST（推荐）
- GET

POST 请求支持的 Content-Type 类型：

- application/json（推荐），必须使用签名方法 v3（TC3-HMAC-SHA256）。
- application/x-www-form-urlencoded，必须使用签名方法 v1（HmacSHA1 或 HmacSHA256）。
- multipart/form-data（仅部分接口支持），必须使用签名方法 v3（TC3-HMAC-SHA256）。

GET 请求的请求包大小不得超过32KB。POST 请求使用签名方法 v1（HmacSHA1、HmacSHA256）时不得超过1MB。POST 请求使用签名方法 v3（TC3-HMAC-SHA256）时支持10MB。

4. 字符编码

均使用 UTF-8 编码。

公共参数

最近更新时间：2024-11-15 02:01:52

公共参数是用于标识用户和接口签名的参数，如非必要，在每个接口单独的文档中不再对这些参数进行说明，但每次请求均需要携带这些参数，才能正常发起请求。

公共参数的具体内容会因您使用的签名方法版本不同而有所差异。

使用签名方法 v3 的公共参数

签名方法 v3（有时也称作 TC3-HMAC-SHA256）相比签名方法 v1（有些文档可能会简称签名方法），更安全，支持更大的请求包，支持 POST JSON 格式，性能有一定提升，推荐使用该签名方法计算签名。完整介绍详见 [签名方法 v3](#)。

注意：出于简化的目的，部分接口文档中的示例使用的是签名方法 v1 GET 请求，而不是更安全的签名方法 v3。

使用签名方法 v3 时，公共参数需要统一放到 HTTP Header 请求头部中，如下表所示：

参数名称	类型	必选	描述
Action	String	是	HTTP 请求头：X-TC-Action。操作的接口名称。取值参考接口文档输入参数章节关于公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。
Region	String	-	HTTP 请求头：X-TC-Region。地域参数，用来标识希望操作哪个地域的数据。取值参考接口文档中输入参数章节关于公共参数 Region 的说明。 注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。
Timestamp	Integer	是	HTTP 请求头：X-TC-Timestamp。当前 UNIX 时间戳，可记录发起 API 请求的时间。例如 1529223702。 注意：如果与服务器时间相差超过5分钟，会引起签名过期错误。
Version	String	是	HTTP 请求头：X-TC-Version。操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
Authorization	String	是	HTTP 标准身份认证头部字段，例如： TC3-HMAC-SHA256 Credential=AKID***/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024 其中， - TC3-HMAC-SHA256：签名方法，目前固定取该值； - Credential：签名凭证，AKID*** 是 SecretId；Date 是 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致；service 为具体产品名，通常为域名前缀。例如，域名 cvm.tencentcloudapi.com 意味着产品名是 cvm。本产品取值为 tag；tc3_request 为固定字符串； - SignedHeaders：参与签名计算的头部信息，content-type 和 host 为必选头部； - Signature：签名摘要，计算过程详见 文档 。
Token	String	否	HTTP 请求头：X-TC-Token。即 安全凭证服务 所颁发的临时安全凭证中的 Token，使用时需要将 SecretId 和 SecretKey 的值替换为临时安全凭证中的 TmpSecretId 和 TmpSecretKey。使用长期密钥时不能设置此 Token 字段。
Language	String	否	HTTP 请求头：X-TC-Language。指定接口返回的语言，仅部分接口支持此参数。取值：zh-CN，en-US。zh-CN 返回中文，en-US 返回英文。

假设用户想要查询广州地域的云服务器实例列表中的前十个，接口参数设置为偏移量 Offset=0，返回数量 Limit=10，则其请求结构按照请求 URL、请求头部、请求体示例如下：

HTTP GET 请求结构示例：

```
https://cvm.tencentcloudapi.com/?Limit=10&Offset=0
```

```
Authorization: TC3-HMAC-SHA256 Credential=AKID*****/2018-10-09/cvm/tc3_request,
SignedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
Content-Type: application/x-www-form-urlencoded
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1539084154
X-TC-Region: ap-guangzhou
```

HTTP POST（application/json）请求结构示例：

```
https://cvm.tencentcloudapi.com/
```

```
Authorization: TC3-HMAC-SHA256 Credential=AKID*****/2018-05-30/cvm/tc3_request,
SignedHeaders=content-type;host, Signature=582c400e06b5924a6f2b5d7d672d79c15b13162d9279b0855cfba6789a8edb4c
Content-Type: application/json
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1527672334
X-TC-Region: ap-guangzhou

{"Offset":0,"Limit":10}
```

HTTP POST（multipart/form-data）请求结构示例（仅特定的接口支持）：

```
https://cvm.tencentcloudapi.com/
```

```
Authorization: TC3-HMAC-SHA256 Credential=AKID*****/2018-05-30/cvm/tc3_request,
SignedHeaders=content-type;host, Signature=582c400e06b5924a6f2b5d7d672d79c15b13162d9279b0855cfba6789a8edb4c
Content-Type: multipart/form-data; boundary=58731222010402
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1527672334
X-TC-Region: ap-guangzhou

--58731222010402
Content-Disposition: form-data; name="Offset"

0
--58731222010402
Content-Disposition: form-data; name="Limit"

10
--58731222010402--
```

使用签名方法 v1 的公共参数

使用签名方法 v1（有时会称作 HmacSHA256 和 HmacSHA1），公共参数需要统一放到请求串中，完整介绍详见[文档](#)

参数名称	类型	必选	描述
Action	String	是	操作的接口名称。取值参考接口文档中输入参数章节关于公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。
Region	String	-	地域参数，用来标识希望操作哪个地域的数据。接口接受的地域取值参考接口文档中输入参数公共参数 Region 的说明。 注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。
Timestamp	Integer	是	当前 UNIX 时间戳，可记录发起 API 请求的时间。例如1529223702，如果与当前时间相差过大，会引起签名过期错误。
Nonce	Integer	是	随机正整数，与 Timestamp 联合起来，用于防止重放攻击。
SecretId	String	是	在 云API密钥 上申请的标识身份的 SecretId，一个 SecretId 对应唯一的 SecretKey，而 SecretKey 会用来生成请求签名 Signature。
Signature	String	是	请求签名，用来验证此次请求的合法性，需要用户根据实际的输入参数计算得出。具体计算方法参见 文档 。
Version	String	是	操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
SignatureMethod	String	否	签名方式，目前支持 HmacSHA256 和 HmacSHA1。只有指定此参数为 HmacSHA256 时，才使用 HmacSHA256 算法验证签名，其他情况均使用 HmacSHA1 验证签名。
Token	String	否	即 安全凭证服务 所颁发的临时安全凭证中的 Token，使用时需要将 SecretId 和 SecretKey 的值替换为临时安全凭证中的 TmpSecretId 和 TmpSecretKey。使用长期密钥时不能设置此 Token 字段。
Language	String	否	指定接口返回的语言，仅部分接口支持此参数。取值：zh-CN，en-US。zh-CN 返回中文，en-US 返回英文。

假设用户想要查询广州地域的云服务器实例列表，其请求结构按照请求 URL、请求头部、请求体示例如下：

HTTP GET 请求结构示例：

```
https://cvm.tencentcloudapi.com/?Action=DescribeInstances&Version=2017-03-12&SignatureMethod=HmacSHA256&Timestamp=1527672334&Signature=37ac2f4fde00b0ac9bd9eadeb459b1bbee224158d66e7ae5fcadb70b2d181d02&Region=ap-guangzhou&Nonce=23823223&SecretId=AKID*****
```

```
Host: cvm.tencentcloudapi.com
```

HTTP POST 请求结构示例：

```
https://cvm.tencentcloudapi.com/
```

```
Host: cvm.tencentcloudapi.com
```

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

```
Action=DescribeInstances&Version=2017-03-12&SignatureMethod=HmacSHA256&Timestamp=1527672334&Signature=37ac2f4fde00b0ac9bd9eadeb459b1bbee224158d66e7ae5fcadb70b2d181d02&Region=ap-guangzhou&Nonce=23823223&SecretId=AKID*****
```

签名方法 v3

最近更新时间：2024-12-25 01:59:16

以下文档说明了签名方法 v3 的签名过程，但仅在您编写自己的代码来调用腾讯云 API 时才有用。我们推荐您使用 [腾讯云 API Explorer](#)，[腾讯云 SDK](#) 和 [腾讯云命令行工具（TCCLI）](#) 等开发者工具，从而无需学习如何对 API 请求进行签名。

推荐使用 API Explorer

<> 点击调试

您可以通过 API Explorer 的【签名串生成】模块查看每个接口签名的生成过程。

腾讯云 API 会对每个请求进行身份验证，用户需要使用安全凭证，经过特定的步骤对请求进行签名（Signature），每个请求都需要在公共参数中指定该签名结果并以指定的方式和格式发送请求。

为什么要进行签名

签名通过以下方式帮助保护请求：

1. 验证请求者的身份

签名确保请求是由持有有效访问密钥的人发送的。请参阅控制台 [云 API 密钥](#) 页面获取密钥相关信息。

2. 保护传输中的数据

为了防止请求在传输过程中被篡改，腾讯云 API 会使用请求参数来计算请求的哈希值，并将生成的哈希值加密后作为请求的一部分，发送到腾讯云 API 服务器。服务器会使用收到的请求参数以同样的过程计算哈希值，并验证请求中的哈希值。如果请求被篡改，将导致哈希值不一致，腾讯云 API 将拒绝本次请求。

签名方法 v3（TC3-HMAC-SHA256）功能上覆盖了以前的签名方法 v1，而且更安全，支持更大的请求，支持 JSON 格式，POST 请求支持传空数组和空字符串，性能有一定提升，推荐使用该签名方法计算签名。

首次接触，建议使用 [API Explorer](#) 中的“签名串生成”功能，选择签名版本为“API 3.0 签名 v3”，可以对生成签名过程进行验证，也可直接生成 SDK 代码。推荐使用腾讯云 API 配套的 8 种常见的编程语言 SDK，已经封装了签名和请求过程，均已开源，支持 [Python](#)、[Java](#)、[PHP](#)、[Go](#)、[NodeJS](#)、[.NET](#)、[C++](#)、[Ruby](#)。

申请安全凭证

本文使用的安全凭证为密钥，密钥包括 SecretId 和 SecretKey。每个用户最多可以拥有两对密钥。

- SecretId：用于标识 API 调用者身份，可以简单类比为用户名。
- SecretKey：用于验证 API 调用者的身份，可以简单类比为密码。
- 用户必须严格保管安全凭证，避免泄露，否则将危及财产安全。如已泄露，请立刻禁用该安全凭证。

申请安全凭证的具体步骤如下：

1. 登录 [腾讯云管理中心控制台](#)。
2. 前往 [云API密钥](#) 的控制台页面。
3. 在 [云API密钥](#) 页面，单击【新建密钥】创建一对密钥。

签名版本 v3 签名过程

云 API 支持 GET 和 POST 请求。对于 GET 方法，只支持 Content-Type: application/x-www-form-urlencoded 协议格式。对于 POST 方法，目前支持 Content-Type: application/json 以及 Content-Type: multipart/form-data 两种协议格式，json 格式绝大多数接口均支持，multipart 格式只有特定接口支持，此时该接口不能使用 json 格式调用，参考具体业务接口文档说明。推荐使用 POST 请求，因为两者的结果并无差异，但 GET 请求只支持 32 KB 以内的请求包。

下面以云服务器查询广州实例列表作为例子，分步骤介绍签名的计算过程。我们选择该接口是因为：

1. 云服务器默认已开通，该接口很常用；

2. 该接口是只读的，不会改变现有资源的状态；
3. 接口覆盖的参数种类较全，可以演示包含数据结构的数组如何使用。

在示例中，不论公共参数或者接口的参数，我们尽量选择容易犯错的情况。在实际调用接口时，请根据实际情况来，每个接口的参数并不相同，不要照抄这个例子的参数和值。此外，这里只展示了部分公共参数和接口输入参数，用户可以根据实际需要添加其他参数，例如 Language 和 Token 公共参数（在 HTTP 头部设置，添加 X-TC- 前缀）。

假设用户的 SecretId 和 SecretKey 分别是：AKID***** 和 *****。用户想查看广州区云服务器名为“未命名”的主机状态，只返回一条数据。则请求可能为：

```
curl -X POST https://cvm.tencentcloudapi.com \
-H "Authorization: TC3-HMAC-SHA256 Credential=AKID*****/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host;x-tc-action, Signature=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f" \
-H "Content-Type: application/json; charset=utf-8" \
-H "Host: cvm.tencentcloudapi.com" \
-H "X-TC-Action: DescribeInstances" \
-H "X-TC-Timestamp: 1551113065" \
-H "X-TC-Version: 2017-03-12" \
-H "X-TC-Region: ap-guangzhou" \
-d '{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}'
```

下面详细解释签名计算过程。

1. 拼接规范请求串

按如下伪代码格式拼接规范请求串（CanonicalRequest）：

```
CanonicalRequest =
HTTPRequestMethod + '\n' +
CanonicalURI + '\n' +
CanonicalQueryString + '\n' +
CanonicalHeaders + '\n' +
SignedHeaders + '\n' +
HashedRequestPayload
```

字段名称	解释
HTTPRequestMethod	HTTP 请求方法（GET、POST）。此示例取值为 POST。
CanonicalURI	URI 参数，API 3.0 固定为正斜杠 (/)。
CanonicalQueryString	发起 HTTP 请求 URL 中的查询字符串，对于 POST 请求，固定为空字符串""，对于 GET 请求，则为 URL 中间号（?）后面的字符串内容，例如：Limit=10&Offset=0。 注意：CanonicalQueryString 需要参考 RFC3986 进行 URLEncode 编码（特殊字符编码后需大写字母），字符集 UTF-8。推荐使用编程语言标准库进行编码。
CanonicalHeaders	参与签名的头部信息，至少包含 host 和 content-type 两个头部，也可加入其他头部参与签名以提高自身请求的唯一性和安全性，此示例额外增加了接口名头部。 拼接规则： 1. 头部 key 和 value 统一转成小写，并去掉首尾空格，按照 key:value\n 格式拼接； 2. 多个头部，按照头部 key（小写）的 ASCII 升序进行拼接。

字段名称	解释
	此示例计算结果是 <code>content-type:application/json; charset=utf-8\nhost:cvm.tencentcloudapi.com\nx-tc-action:describeinstances\n</code>。 注意：<code>content-type</code> 必须和实际发送的相符合，有些编程语言网络库即使未指定也会自动添加 <code>charset</code> 值，如果签名时和发送时不一致，服务器会返回签名校验失败。
SignedHeaders	参与签名的头部信息，说明此次请求有哪些头部参与了签名，和 CanonicalHeaders 包含的头部内容是一一对应的。<code>content-type</code> 和 <code>host</code> 为必选头部。 拼接规则： 1. 头部 key 统一转成小写； 2. 多个头部 key（小写）按照 ASCII 升序进行拼接，并且以分号（;）分隔。 此示例为 <code>content-type;host;x-tc-action</code>
HashedRequestPayload	请求正文（payload，即 body，此示例为 <code>{"Limit": 1, "Filters": [{"Values": [{"u672a\u547d\u540d"}], "Name": "instance-name"}]}</code>）的哈希值，计算伪代码为 <code>Lowercase(HexEncode(Hash.SHA256(RequestPayload)))</code>，即对 HTTP 请求正文做 SHA256 哈希，然后十六进制编码，最后编码串转换成小写字母。对于 GET 请求，RequestPayload 固定为空字符串。此示例计算结果是 <code>35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064</code>。

根据以上规则，示例中得到的规范请求串如下：

```
POST
/

content-type:application/json; charset=utf-8
host:cvm.tencentcloudapi.com
x-tc-action:describeinstances

content-type;host;x-tc-action
35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064
```

2. 拼接待签名字符串

按如下格式拼接待签名字符串：

```
StringToSign =
Algorithm + "\n" +
RequestTimestamp + "\n" +
CredentialScope + "\n" +
HashedCanonicalRequest
```

字段名称	解释
Algorithm	签名算法，目前固定为 <code>TC3-HMAC-SHA256</code> 。
RequestTimestamp	请求时间戳，即请求头部的公共参数 <code>X-TC-Timestamp</code> 取值，取当前时间 UNIX 时间戳，精确到秒。此示例取值为 <code>1551113065</code> 。
CredentialScope	凭证范围，格式为 <code>Date/service/tc3_request</code> ，包含日期、所请求的服务和终止字符串（ <code>tc3_request</code> ）。 Date 为 UTC 标准时间的日期，取值需要和公共参数 <code>X-TC-Timestamp</code> 换算的 UTC 标准时间日期一致； service 为产品名，必须与调用的产品域名一致。此示例计算结果是 <code>2019-02-25/cvm/tc3_request</code> 。

字段名称	解释
HashedCanonicalRequest	前述步骤拼接所得规范请求串的哈希值，计算伪代码为 <code>Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))</code> 。此示例计算结果是 7019a55be8395899b900fb5564e4200d984910f34794a27cb3fb7d10ff6a1e84。

注意：

1. Date 必须从时间戳 X-TC-Timestamp 计算得到，且时区为 UTC+0。如果加入系统本地时区信息，例如东八区，将导致白天和晚上调用成功，但是凌晨时调用必定失败。假设时间戳为 1551113065，在东八区的时间是 2019-02-26 00:44:25，但是计算得到的 Date 取 UTC+0 的日期应为 2019-02-25，而不是 2019-02-26。
2. Timestamp 必须是当前系统时间，且需确保系统时间和标准时间是同步的，如果相差超过五分钟则必定失败。如果长时间不和标准时间同步，可能运行一段时间后，请求失败，返回签名过期错误。

根据以上规则，示例中得到的待签名字符串如下：

```
TC3-HMAC-SHA256
1551113065
2019-02-25/cvm/tc3_request
7019a55be8395899b900fb5564e4200d984910f34794a27cb3fb7d10ff6a1e84
```

3. 计算签名

1) 计算派生签名密钥，伪代码如下：

```
SecretKey = "*****"
SecretDate = HMAC_SHA256("TC3" + SecretKey, Date)
SecretService = HMAC_SHA256(SecretDate, Service)
SecretSigning = HMAC_SHA256(SecretService, "tc3_request")
```

派生出的密钥 SecretDate、SecretService 和 SecretSigning 是二进制的的数据，可能包含不可打印字符，将其转为十六进制字符串打印的输出分别为：da98fb70dcf6b112dc21038d1eeeb3a95c74b4dcb12c1131f864f6066bd02be0，8d70cbefb03939f929db64d32dc2ba89b1095620119fe3e050e2b18c5bd2752f，b596b923aad85185e2d1f6659d2a062e0a86731226e021e61bfe06f7ed05f5af。

请注意，不同的编程语言，HMAC 库函数中参数顺序可能不一样，请以实际情况为准。此处的伪代码密钥参数 key 在前，消息参数 data 在后。通常标准库函数会提供二进制格式的返回值，也可能会提供打印友好的十六进制格式的返回值，此处使用的是二进制格式。

字段名称	解释
SecretKey	原始的 SecretKey，即 *****。
Date	即 Credential 中的 Date 字段信息。此示例取值为 2019-02-25。
Service	即 Credential 中的 Service 字段信息。此示例取值为 cvm。

2) 计算签名，伪代码如下：

```
Signature = HexEncode(HMAC_SHA256(SecretSigning, StringToSign))
```

此示例计算结果是 10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f。

4. 拼接 Authorization

按如下格式拼接 Authorization：

```
Authorization =
Algorithm + ' ' +
'Credential=' + SecretId + '/' + CredentialScope + ', ' +
'SignedHeaders=' + SignedHeaders + ', ' +
'Signature=' + Signature
```

字段名称	解释
Algorithm	签名方法，固定为 TC3-HMAC-SHA256。
SecretId	密钥对中的 SecretId，即 AKID*****。
CredentialScope	见上文，凭证范围。此示例计算结果是 2019-02-25/cvm/tc3_request。
SignedHeaders	见上文，参与签名的头部信息。此示例取值为 content-type;host;x-tc-action。
Signature	签名值。此示例计算结果是 10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f。

根据以上规则，示例中得到的值为：

```
TC3-HMAC-SHA256 Credential=AKID*****/2019-02-25/cvm/tc3_request, SignedHeaders=c
ontent-type;host;x-tc-action, Signature=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f
```

最终完整的调用信息如下：

```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKID*****/2019-02-25/cvm/tc3_request,
SignedHeaders=content-type;host;x-tc-action, Signature=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b4
6093f6ab6c4f
Content-Type: application/json; charset=utf-8
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1551113065
X-TC-Region: ap-guangzhou

{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}
```

⚠ 注意：

请求发送时的 HTTP 头部（Header）和请求体（Payload）必须和签名计算过程中的内容完全一致，否则会返回签名不一致错误。可以通过打印实际请求内容，网络抓包等方式对比排查。

签名演示

在实际调用 API 3.0 时，推荐使用配套的腾讯云 SDK 3.0，SDK 封装了签名的过程，开发时只关注产品提供的具体接口即可。详细信息参见 [SDK 中心](#)。当前支持的编程语言有：

- [Python](#)
- [Java](#)
- [PHP](#)
- [Go](#)
- [NodeJS](#)
- [.NET](#)
- [C++](#)
- [Ruby](#)

下面提供了不同产品的生成签名 demo，您可以找到对应的产品参考签名的生成：

- [Signature Demo](#)

为了更清楚地解释签名过程，下面以实际编程语言为例，将上述的签名过程完整实现。请求的域名、调用的接口和参数的取值都以上述签名过程为准，代码只为解释签名过程，并不具备通用性，实际开发请尽量使用 SDK。

Java

```
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimeZone;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DataConverter;

public class TencentCloudAPITC3Demo {
    private final static Charset UTF8 = StandardCharsets.UTF_8;
    // 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****
    private final static String SECRET_ID = System.getenv("TENCENTCLOUD_SECRET_ID");
    // 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
    private final static String SECRET_KEY = System.getenv("TENCENTCLOUD_SECRET_KEY");
    private final static String CT_JSON = "application/json; charset=utf-8";

    public static byte[] hmac256(byte[] key, String msg) throws Exception {
        Mac mac = Mac.getInstance("HmacSHA256");
        SecretKeySpec secretKeySpec = new SecretKeySpec(key, mac.getAlgorithm());
        mac.init(secretKeySpec);
        return mac.doFinal(msg.getBytes(UTF8));
    }

    public static String sha256Hex(String s) throws Exception {
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[] d = md.digest(s.getBytes(UTF8));
    }
```

```
return DatatypeConverter.printHexBinary(d).toLowerCase();
}

public static void main(String[] args) throws Exception {
    String service = "cvm";
    String host = "cvm.tencentcloudapi.com";
    String region = "ap-guangzhou";
    String action = "DescribeInstances";
    String version = "2017-03-12";
    String algorithm = "TC3-HMAC-SHA256";
    String timestamp = "1551113065";
    //String timestamp = String.valueOf(System.currentTimeMillis() / 1000);
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
    // 注意时区, 否则容易出错
    sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
    String date = sdf.format(new Date(Long.valueOf(timestamp + "000")));

    // ***** 步骤 1: 拼接规范请求串 *****
    String httpRequestMethod = "POST";
    String canonicalUri = "/";
    String canonicalQueryString = "";
    String canonicalHeaders = "content-type:application/json; charset=utf-8\n"
+ "host:" + host + "\n" + "x-tc-action:" + action.toLowerCase() + "\n";
    String signedHeaders = "content-type;host;x-tc-action";

    String payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name\"}] }";
    String hashedRequestPayload = sha256Hex(payload);
    String canonicalRequest = httpRequestMethod + "\n" + canonicalUri + "\n" + canonicalQueryString + "\n"
+ canonicalHeaders + "\n" + signedHeaders + "\n" + hashedRequestPayload;
    System.out.println(canonicalRequest);

    // ***** 步骤 2: 拼接待签名字符串 *****
    String credentialScope = date + "/" + service + "/" + "tc3_request";
    String hashedCanonicalRequest = sha256Hex(canonicalRequest);
    String stringToSign = algorithm + "\n" + timestamp + "\n" + credentialScope + "\n" + hashedCanonicalRequest;
    System.out.println(stringToSign);

    // ***** 步骤 3: 计算签名 *****
    byte[] secretDate = hmac256(("TC3" + SECRET_KEY).getBytes(UTF8), date);
    byte[] secretService = hmac256(secretDate, service);
    byte[] secretSigning = hmac256(secretService, "tc3_request");
    String signature = DatatypeConverter.printHexBinary(hmac256(secretSigning, stringToSign)).toLowerCase();
    System.out.println(signature);

    // ***** 步骤 4: 拼接 Authorization *****
    String authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + credentialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
    System.out.println(authorization);
}
```

```
TreeMap<String, String> headers = new TreeMap<String, String>();
headers.put("Authorization", authorization);
headers.put("Content-Type", CT_JSON);
headers.put("Host", host);
headers.put("X-TC-Action", action);
headers.put("X-TC-Timestamp", timestamp);
headers.put("X-TC-Version", version);
headers.put("X-TC-Region", region);

StringBuilder sb = new StringBuilder();
sb.append("curl -X POST https://").append(host)
.append(" -H \"Authorization: ").append(authorization).append("\")")
.append(" -H \"Content-Type: application/json; charset=utf-8\"")
.append(" -H \"Host: ").append(host).append("\")")
.append(" -H \"X-TC-Action: ").append(action).append("\")")
.append(" -H \"X-TC-Timestamp: ").append(timestamp).append("\")")
.append(" -H \"X-TC-Version: ").append(version).append("\")")
.append(" -H \"X-TC-Region: ").append(region).append("\")")
.append(" -d '").append(payload).append("'");
System.out.println(sb.toString());
}
}
```

Python

```
# -*- coding: utf-8 -*-
import hashlib, hmac, json, os, sys, time
from datetime import datetime

# 密钥参数
# 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****
secret_id = os.environ.get("TENCENTCLOUD_SECRET_ID")
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
secret_key = os.environ.get("TENCENTCLOUD_SECRET_KEY")

service = "cvm"
host = "cvm.tencentcloudapi.com"
endpoint = "https://" + host
region = "ap-guangzhou"
action = "DescribeInstances"
version = "2017-03-12"
algorithm = "TC3-HMAC-SHA256"
#timestamp = int(time.time())
timestamp = 1551113065
date = datetime.utcfromtimestamp(timestamp).strftime("%Y-%m-%d")
params = {"Limit": 1, "Filters": [{"Values": [u"未命名"], "Name": "instance-name"}]}

# ***** 步骤 1: 拼接规范请求串 *****
```

```
http_request_method = "POST"
canonical_uri = "/"
canonical_querystring = ""
ct = "application/json; charset=utf-8"
payload = json.dumps(params)
canonical_headers = "content-type:%s\nhost:%s\nx-tc-action:%s\n" % (ct, host, action.lower())
signed_headers = "content-type;host;x-tc-action"
hashed_request_payload = hashlib.sha256(payload.encode("utf-8")).hexdigest()
canonical_request = (http_request_method + "\n" +
canonical_uri + "\n" +
canonical_querystring + "\n" +
canonical_headers + "\n" +
signed_headers + "\n" +
hashed_request_payload)
print(canonical_request)

# ***** 步骤 2：拼接待签名字符串 *****
credential_scope = date + "/" + service + "/" + "tc3_request"
hashed_canonical_request = hashlib.sha256(canonical_request.encode("utf-8")).hexdigest()
string_to_sign = (algorithm + "\n" +
str(timestamp) + "\n" +
credential_scope + "\n" +
hashed_canonical_request)
print(string_to_sign)

# ***** 步骤 3：计算签名 *****
# 计算签名摘要函数
def sign(key, msg):
return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()
secret_date = sign(("TC3" + secret_key).encode("utf-8"), date)
secret_service = sign(secret_date, service)
secret_signing = sign(secret_service, "tc3_request")
signature = hmac.new(secret_signing, string_to_sign.encode("utf-8"), hashlib.sha256).hexdigest()
print(signature)

# ***** 步骤 4：拼接 Authorization *****
authorization = (algorithm + " " +
"Credential=" + secret_id + "/" + credential_scope + ", " +
"SignedHeaders=" + signed_headers + ", " +
"Signature=" + signature)
print(authorization)

print('curl -X POST ' + endpoint
+ ' -H "Authorization: ' + authorization + '" '
+ ' -H "Content-Type: application/json; charset=utf-8" '
+ ' -H "Host: ' + host + '" '
+ ' -H "X-TC-Action: ' + action + '" '
+ ' -H "X-TC-Timestamp: ' + str(timestamp) + '" '
+ ' -H "X-TC-Version: ' + version + '" ')
```

```
+ ' -H "X-TC-Region: ' + region + '"'  
+ " -d '" + payload + '"")
```

Golang

```
package main  
  
import (  
    "crypto/hmac"  
    "crypto/sha256"  
    "encoding/hex"  
    "fmt"  
    "os"  
    "strings"  
    "time"  
)  
  
func sha256hex(s string) string {  
    b := sha256.Sum256([]byte(s))  
    return hex.EncodeToString(b[:])  
}  
  
func hmacsha256(s, key string) string {  
    hashed := hmac.New(sha256.New, []byte(key))  
    hashed.Write([]byte(s))  
    return string(hashed.Sum(nil))  
}  
  
func main() {  
    // 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****  
    secretId := os.Getenv("TENCENTCLOUD_SECRET_ID")  
    // 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****  
    secretKey := os.Getenv("TENCENTCLOUD_SECRET_KEY")  
    host := "cvm.tencentcloudapi.com"  
    algorithm := "TC3-HMAC-SHA256"  
    service := "cvm"  
    version := "2017-03-12"  
    action := "DescribeInstances"  
    region := "ap-guangzhou"  
    //var timestamp int64 = time.Now().Unix()  
    var timestamp int64 = 1551113065  
  
    // step 1: build canonical request string  
    httpRequestMethod := "POST"  
    canonicalURI := "/"  
    canonicalQueryString := ""  
    canonicalHeaders := fmt.Sprintf("content-type:%s\nhost:%s\nx-tc-action:%s\n",  
        "application/json; charset=utf-8", host, strings.ToLower(action))  
    signedHeaders := "content-type;host;x-tc-action"
```

```
payload := `{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}`
hashedRequestPayload := sha256hex(payload)
canonicalRequest := fmt.Sprintf("%s\n%s\n%s\n%s\n%s\n%s\n%s",
    httpRequestMethod,
    canonicalURI,
    canonicalQueryString,
    canonicalHeaders,
    signedHeaders,
    hashedRequestPayload)
fmt.Println(canonicalRequest)

// step 2: build string to sign
date := time.Unix(timestamp, 0).UTC().Format("2006-01-02")
credentialScope := fmt.Sprintf("%s/%s/tc3_request", date, service)
hashedCanonicalRequest := sha256hex(canonicalRequest)
string2sign := fmt.Sprintf("%s\n%d\n%s\n%s",
    algorithm,
    timestamp,
    credentialScope,
    hashedCanonicalRequest)
fmt.Println(string2sign)

// step 3: sign string
secretDate := hmacsha256(date, "TC3"+secretKey)
secretService := hmacsha256(service, secretDate)
secretSigning := hmacsha256("tc3_request", secretService)
signature := hex.EncodeToString([]byte(hmacsha256(string2sign, secretSigning)))
fmt.Println(signature)

// step 4: build authorization
authorization := fmt.Sprintf("%s Credential=%s/%s, SignedHeaders=%s, Signature=%s",
    algorithm,
    secretId,
    credentialScope,
    signedHeaders,
    signature)
fmt.Println(authorization)

curl := fmt.Sprintf(`curl -X POST https://%s\
-H "Authorization: %s"\
-H "Content-Type: application/json; charset=utf-8"\
-H "Host: %s" -H "X-TC-Action: %s"\
-H "X-TC-Timestamp: %d"\
-H "X-TC-Version: %s"\
-H "X-TC-Region: %s"\
-d '%s'`, host, authorization, host, action, timestamp, version, region, payload)
fmt.Println(curl)
}
```

PHP

```
<?php
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****
$secretId = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
$secretKey = getenv("TENCENTCLOUD_SECRET_KEY");
$host = "cvm.tencentcloudapi.com";
$service = "cvm";
$version = "2017-03-12";
$action = "DescribeInstances";
$region = "ap-guangzhou";
// $timestamp = time();
$timestamp = 1551113065;
$algorithm = "TC3-HMAC-SHA256";

// step 1: build canonical request string
$httpRequestMethod = "POST";
$canonicalUri = "/";
$canonicalQueryString = "";
$canonicalHeaders = implode("\n", [
    "content-type:application/json; charset=utf-8",
    "host:".$host,
    "x-tc-action:".strtolower($action),
    ""
]);
$signedHeaders = implode(";", [
    "content-type",
    "host",
    "x-tc-action",
]);
$payload = '{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}';
$hashedRequestPayload = hash("SHA256", $payload);
$canonicalRequest = $httpRequestMethod."\n"
.$canonicalUri."\n"
.$canonicalQueryString."\n"
.$canonicalHeaders."\n"
.$signedHeaders."\n"
.$hashedRequestPayload;
echo $canonicalRequest.PHP_EOL;

// step 2: build string to sign
$date = gmdate("Y-m-d", $timestamp);
$credentialScope = $date."/".$service."/tc3_request";
$hashedCanonicalRequest = hash("SHA256", $canonicalRequest);
$stringToSign = $algorithm."\n"
.$timestamp."\n"
.$credentialScope."\n"
.$hashedCanonicalRequest;
echo $stringToSign.PHP_EOL;
```

```
// step 3: sign string
$secretDate = hash_hmac("SHA256", $date, "TC3".$secretKey, true);
$secretService = hash_hmac("SHA256", $service, $secretDate, true);
$secretSigning = hash_hmac("SHA256", "tc3_request", $secretService, true);
$signature = hash_hmac("SHA256", $stringToSign, $secretSigning);
echo $signature.PHP_EOL;

// step 4: build authorization
$authorization = $algorithm
." Credential=".$secretId."/".$credentialScope
.", SignedHeaders=".$signedHeaders.", Signature=".$signature;
echo $authorization.PHP_EOL;

$curl = "curl -X POST https://" . $host
.' -H "Authorization: ' . $authorization . '"'
.' -H "Content-Type: application/json; charset=utf-8"'
.' -H "Host: ' . $host . '"'
.' -H "X-TC-Action: ' . $action . '"'
.' -H "X-TC-Timestamp: ' . $timestamp . '"'
.' -H "X-TC-Version: ' . $version . '"'
.' -H "X-TC-Region: ' . $region . '"'
." -d '" . $payload . "'";
echo $curl.PHP_EOL;
```

Ruby

```
# -*- coding: UTF-8 -*-
# require ruby>=2.3.0
require 'digest'
require 'json'
require 'time'
require 'openssl'

# 密钥参数
# 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****
secret_id = ENV["TENCENTCLOUD_SECRET_ID"]
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
secret_key = ENV["TENCENTCLOUD_SECRET_KEY"]

service = 'cvm'
host = 'cvm.tencentcloudapi.com'
endpoint = 'https://' + host
region = 'ap-guangzhou'
action = 'DescribeInstances'
version = '2017-03-12'
algorithm = 'TC3-HMAC-SHA256'
# timestamp = Time.now.to_i
timestamp = 1551113065
```

```
date = Time.at(timestamp).utc.strftime('%Y-%m-%d')

# ***** 步骤 1: 拼接规范请求串 *****

http_request_method = 'POST'
canonical_uri = '/'
canonical_querystring = ''
canonical_headers = "content-type:application/json; charset=utf-8\nhost:#{host}\nx-tc-action:#{action.downcase}\n"
signed_headers = 'content-type;host;x-tc-action'
# params = { 'Limit' => 1, 'Filters' => [{ 'Name' => 'instance-name', 'Values' => ['未命名'] }] }
# payload = JSON.generate(params, { 'ascii_only' => true, 'space' => ' ' })
# json will generate in random order, to get specified result in example, we hard-code it here.
payload = '{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}'
hashed_request_payload = Digest::SHA256.hexdigest(payload)
canonical_request = [
  http_request_method,
  canonical_uri,
  canonical_querystring,
  canonical_headers,
  signed_headers,
  hashed_request_payload,
].join("\n")

puts canonical_request

# ***** 步骤 2: 拼接待签名字符串 *****

credential_scope = date + '/' + service + '/' + 'tc3_request'
hashed_request_payload = Digest::SHA256.hexdigest(canonical_request)
string_to_sign = [
  algorithm,
  timestamp.to_s,
  credential_scope,
  hashed_request_payload,
].join("\n")
puts string_to_sign

# ***** 步骤 3: 计算签名 *****

digest = OpenSSL::Digest.new('sha256')
secret_date = OpenSSL::HMAC.digest(digest, 'TC3' + secret_key, date)
secret_service = OpenSSL::HMAC.digest(digest, secret_date, service)
secret_signing = OpenSSL::HMAC.digest(digest, secret_service, 'tc3_request')
signature = OpenSSL::HMAC.hexdigest(digest, secret_signing, string_to_sign)
puts signature

# ***** 步骤 4: 拼接 Authorization *****

authorization = "#{algorithm} Credential=#{secret_id}/#{credential_scope}, SignedHeaders=#{signed_headers}, Signature=#{signature}"
puts authorization

puts 'curl -X POST ' + endpoint \
```

```
+ ' -H "Authorization: ' + authorization + "' \
+ ' -H "Content-Type: application/json; charset=utf-8"' \
+ ' -H "Host: ' + host + "' \
+ ' -H "X-TC-Action: ' + action + "' \
+ ' -H "X-TC-Timestamp: ' + timestamp.to_s + "' \
+ ' -H "X-TC-Version: ' + version + "' \
+ ' -H "X-TC-Region: ' + region + "' \
+ " -d '" + payload + "'"
```

DotNet

```
using System;
using System.Collections.Generic;
using System.Security.Cryptography;
using System.Text;

public class Application
{
    public static string SHA256Hex(string s)
    {
        using (SHA256 algo = SHA256.Create())
        {
            byte[] hashbytes = algo.ComputeHash(Encoding.UTF8.GetBytes(s));
            StringBuilder builder = new StringBuilder();
            for (int i = 0; i < hashbytes.Length; ++i)
            {
                builder.Append(hashbytes[i].ToString("x2"));
            }
            return builder.ToString();
        }
    }

    public static byte[] HmacSHA256(byte[] key, byte[] msg)
    {
        using (HMACSHA256 mac = new HMACSHA256(key))
        {
            return mac.ComputeHash(msg);
        }
    }

    public static Dictionary<String, String> BuildHeaders(string secretid,
        string secretkey, string service, string endpoint, string region,
        string action, string version, DateTime date, string requestPayload)
    {
        string datestr = date.ToString("yyyy-MM-dd");
        DateTime startTime = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc);
        long requestTimestamp = (long)Math.Round((date - startTime).TotalMilliseconds, MidpointRounding.AwayFromZero) / 1000;
        // ***** 步骤 1: 拼接规范请求串 *****
    }
}
```

```
string algorithm = "TC3-HMAC-SHA256";
string httpRequestMethod = "POST";
string canonicalUri = "/";
string canonicalQueryString = "";
string contentType = "application/json";
string canonicalHeaders = "content-type:" + contentType + "; charset=utf-8\n"
+ "host:" + endpoint + "\n"
+ "x-tc-action:" + action.ToLower() + "\n";
string signedHeaders = "content-type;host;x-tc-action";
string hashedRequestPayload = SHA256Hex(requestPayload);
string canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload;
Console.WriteLine(canonicalRequest);

// ***** 步骤 2: 拼接待签名字符串 *****
string credentialScope = datestr + "/" + service + "/" + "tc3_request";
string hashedCanonicalRequest = SHA256Hex(canonicalRequest);
string stringToSign = algorithm + "\n"
+ requestTimestamp.ToString() + "\n"
+ credentialScope + "\n"
+ hashedCanonicalRequest;
Console.WriteLine(stringToSign);

// ***** 步骤 3: 计算签名 *****
byte[] tc3SecretKey = Encoding.UTF8.GetBytes("TC3" + secretkey);
byte[] secretDate = HmacSHA256(tc3SecretKey, Encoding.UTF8.GetBytes(datestr));
byte[] secretService = HmacSHA256(secretDate, Encoding.UTF8.GetBytes(service));
byte[] secretSigning = HmacSHA256(secretService, Encoding.UTF8.GetBytes("tc3_request"));
byte[] signatureBytes = HmacSHA256(secretSigning, Encoding.UTF8.GetBytes(stringToSign));
string signature = BitConverter.ToString(signatureBytes).Replace("-", "").ToLower();
Console.WriteLine(signature);

// ***** 步骤 4: 拼接 Authorization *****
string authorization = algorithm + " "
+ "Credential=" + secretid + "/" + credentialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", "
+ "Signature=" + signature;
Console.WriteLine(authorization);

Dictionary<string, string> headers = new Dictionary<string, string>();
headers.Add("Authorization", authorization);
headers.Add("Host", endpoint);
headers.Add("Content-Type", contentType + "; charset=utf-8");
headers.Add("X-TC-Timestamp", requestTimestamp.ToString());
headers.Add("X-TC-Version", version);
headers.Add("X-TC-Action", action);
```

```
headers.Add("X-TC-Region", region);
return headers;
}

public static void Main(string[] args)
{
    // 密钥参数
    // 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****
    string SECRET_ID = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_ID");
    // 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
    string SECRET_KEY = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_KEY");

    string service = "cvm";
    string endpoint = "cvm.tencentcloudapi.com";
    string region = "ap-guangzhou";
    string action = "DescribeInstances";
    string version = "2017-03-12";

    // 此处由于示例规范的原因, 采用时间戳2019-02-26 00:44:25, 此参数作为示例, 如果在项目中, 您应当使用:
    // DateTime date = DateTime.UtcNow;
    // 注意时区, 建议此时间统一采用UTC时间戳, 否则容易出错
    DateTime date = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc).AddSeconds(1551113065);
    string requestPayload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name\"}] }";

    Dictionary<string, string> headers = BuildHeaders(SECRET_ID, SECRET_KEY, service
, endpoint, region, action, version, date, requestPayload);

    Console.WriteLine("POST https://cvm.tencentcloudapi.com");
    foreach (KeyValuePair<string, string> kv in headers)
    {
        Console.WriteLine(kv.Key + ": " + kv.Value);
    }
    Console.WriteLine();
    Console.WriteLine(requestPayload);
}
}
```

NodeJS

```
const crypto = require('crypto');

function sha256(message, secret = '', encoding) {
    const hmac = crypto.createHmac('sha256', secret)
    return hmac.update(message).digest(encoding)
}

function getHash(message, encoding = 'hex') {
    const hash = crypto.createHash('sha256')
    return hash.update(message).digest(encoding)
}
```

```
}

function getDate(timestamp) {
  const date = new Date(timestamp * 1000)
  const year = date.getUTCFullYear()
  const month = ('0' + (date.getUTCMonth() + 1)).slice(-2)
  const day = ('0' + date.getUTCDate()).slice(-2)
  return `${year}-${month}-${day}`
}

function main(){
  // 密钥参数
  // 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****
  const SECRET_ID = process.env.TENCENTCLOUD_SECRET_ID
  // 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
  const SECRET_KEY = process.env.TENCENTCLOUD_SECRET_KEY

  const endpoint = "cvm.tencentcloudapi.com"
  const service = "cvm"
  const region = "ap-guangzhou"
  const action = "DescribeInstances"
  const version = "2017-03-12"
  //const timestamp = getTime()
  const timestamp = 1551113065
  //时间处理, 获取世界时间日期
  const date = getDate(timestamp)

  // ***** 步骤 1: 拼接规范请求串 *****
  const payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name\"}]}"

  const hashedRequestPayload = getHash(payload);
  const httpRequestMethod = "POST"
  const canonicalUri = "/"
  const canonicalQueryString = ""
  const canonicalHeaders = "content-type:application/json; charset=utf-8\n"
  + "host:" + endpoint + "\n"
  + "x-tc-action:" + action.toLowerCase() + "\n"
  const signedHeaders = "content-type;host;x-tc-action"

  const canonicalRequest = httpRequestMethod + "\n"
  + canonicalUri + "\n"
  + canonicalQueryString + "\n"
  + canonicalHeaders + "\n"
  + signedHeaders + "\n"
  + hashedRequestPayload
  console.log(canonicalRequest)

  // ***** 步骤 2: 拼接待签名字符串 *****
  const algorithm = "TC3-HMAC-SHA256"
```

```
const hashedCanonicalRequest = getHash(canonicalRequest);
const credentialScope = date + "/" + service + "/" + "tc3_request"
const stringToSign = algorithm + "\n" +
timestamp + "\n" +
credentialScope + "\n" +
hashedCanonicalRequest
console.log(stringToSign)

// ***** 步骤 3: 计算签名 *****
const kDate = sha256(date, 'TC3' + SECRET_KEY)
const kService = sha256(service, kDate)
const kSigning = sha256('tc3_request', kService)
const signature = sha256(stringToSign, kSigning, 'hex')
console.log(signature)

// ***** 步骤 4: 拼接 Authorization *****
const authorization = algorithm + " " +
"Credential=" + SECRET_ID + "/" + credentialScope + ", " +
"SignedHeaders=" + signedHeaders + ", " +
"Signature=" + signature
console.log(authorization)

const curlcmd = 'curl -X POST ' + "https://" + endpoint
+ ' -H "Authorization: ' + authorization + '"'
+ ' -H "Content-Type: application/json; charset=utf-8"'
+ ' -H "Host: ' + endpoint + '"'
+ ' -H "X-TC-Action: ' + action + '"'
+ ' -H "X-TC-Timestamp: ' + timestamp.toString() + '"'
+ ' -H "X-TC-Version: ' + version + '"'
+ ' -H "X-TC-Region: ' + region + '"'
+ " -d '" + payload + '"'
console.log(curlcmd)
}
main()
```

C++

```
#include <algorithm>
#include <cstdlib>
#include <iostream>
#include <iomanip>
#include <sstream>
#include <string>
#include <stdio.h>
#include <time.h>
#include <openssl/sha.h>
#include <openssl/hmac.h>

using namespace std;
```

```
string get_data(int64_t &timestamp)
{
    string utcDate;
    char buff[20] = {0};
    // time_t timenow;
    struct tm sttime;
    sttime = *gmtime(&timestamp);
    strftime(buff, sizeof(buff), "%Y-%m-%d", &sttime);
    utcDate = string(buff);
    return utcDate;
}

string int2str(int64_t n)
{
    std::stringstream ss;
    ss << n;
    return ss.str();
}

string sha256Hex(const string &str)
{
    char buf[3];
    unsigned char hash[SHA256_DIGEST_LENGTH];
    SHA256_CTX sha256;
    SHA256_Init(&sha256);
    SHA256_Update(&sha256, str.c_str(), str.size());
    SHA256_Final(hash, &sha256);
    std::string NewString = "";
    for(int i = 0; i < SHA256_DIGEST_LENGTH; i++)
    {
        sprintf(buf, sizeof(buf), "%02x", hash[i]);
        NewString = NewString + buf;
    }
    return NewString;
}

string HmacSha256(const string &key, const string &input)
{
    unsigned char hash[32];

    HMAC_CTX *h;
    #if OPENSSL_VERSION_NUMBER < 0x10100000L
    HMAC_CTX hmac;
    HMAC_CTX_init(&hmac);
    h = &hmac;
    #else
    h = HMAC_CTX_new();
    #endif
```

```
HMAC_Init_ex(h, &key[0], key.length(), EVP_sha256(), NULL);
HMAC_Update(h, ( unsigned char* )&input[0], input.length());
unsigned int len = 32;
HMAC_Final(h, hash, &len);

#ifdef OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX_cleanup(h);
#else
HMAC_CTX_free(h);
#endif

std::stringstream ss;
ss << std::setfill('0');
for (int i = 0; i < len; i++)
{
    ss << hash[i];
}

return (ss.str());
}

string HexEncode(const string &input)
{
    static const char* const lut = "0123456789abcdef";
    size_t len = input.length();

    string output;
    output.reserve(2 * len);
    for (size_t i = 0; i < len; ++i)
    {
        const unsigned char c = input[i];
        output.push_back(lut[c >> 4]);
        output.push_back(lut[c & 15]);
    }
    return output;
}

int main()
{
    // 密钥参数
    // 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****
    string SECRET_ID = getenv("TENCENTCLOUD_SECRET_ID");
    // 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
    string SECRET_KEY = getenv("TENCENTCLOUD_SECRET_KEY");

    string service = "cvm";
    string host = "cvm.tencentcloudapi.com";
    string region = "ap-guangzhou";
    string action = "DescribeInstances";
    string version = "2017-03-12";
```

```
int64_t timestamp = 1551113065;
string date = get_data(timestamp);

// ***** 步骤 1: 拼接规范请求串 *****
string httpRequestMethod = "POST";
string canonicalUri = "/";
string canonicalQueryString = "";
string lower = action;
std::transform(action.begin(), action.end(), lower.begin(), ::tolower);
string canonicalHeaders = string("content-type:application/json; charset=utf-8\n")
+ "host:" + host + "\n"
+ "x-tc-action:" + lower + "\n";
string signedHeaders = "content-type;host;x-tc-action";
string payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name\"}] }";
string hashedRequestPayload = sha256Hex(payload);
string canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload;
cout << canonicalRequest << endl;

// ***** 步骤 2: 拼接待签名字符串 *****
string algorithm = "TC3-HMAC-SHA256";
string RequestTimestamp = int2str(timestamp);
string credentialScope = date + "/" + service + "/" + "tc3_request";
string hashedCanonicalRequest = sha256Hex(canonicalRequest);
string stringToSign = algorithm + "\n" + RequestTimestamp + "\n" + credentialScope + "\n" + hashedCanonicalRequest;
cout << stringToSign << endl;

// ***** 步骤 3: 计算签名 *****
string kKey = "TC3" + SECRET_KEY;
string kDate = HmacSha256(kKey, date);
string kService = HmacSha256(kDate, service);
string kSigning = HmacSha256(kService, "tc3_request");
string signature = HexEncode(HmacSha256(kSigning, stringToSign));
cout << signature << endl;

// ***** 步骤 4: 拼接 Authorization *****
string authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + credentialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
cout << authorization << endl;

string curlcmd = "curl -X POST https://" + host + "\n"
+ " -H \"Authorization: \" + authorization + "\"\n"
+ " -H \"Content-Type: application/json; charset=utf-8\" + "\n"
+ " -H \"Host: \" + host + "\"\n"
```

```
+ " -H \"X-TC-Action: \" + action + "\"\\n"
+ " -H \"X-TC-Timestamp: \" + RequestTimestamp + "\"\\n"
+ " -H \"X-TC-Version: \" + version + "\"\\n"
+ " -H \"X-TC-Region: \" + region + "\"\\n"
+ " -d '" + payload + "'";
cout << curlcmd << endl;
return 0;
};
```

C

```
#include <ctype.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdint.h>
#include <openssl/sha.h>
#include <openssl/hmac.h>

void get_utc_date(int64_t timestamp, char* utc, int len)
{
    // time_t timenow;
    struct tm sttime;
    sttime = *gmtime(&timestamp);
    strftime(utc, len, "%Y-%m-%d", &sttime);
}

void sha256_hex(const char* str, char* result)
{
    char buf[3];
    unsigned char hash[SHA256_DIGEST_LENGTH];
    SHA256_CTX sha256;
    SHA256_Init(&sha256);
    SHA256_Update(&sha256, str, strlen(str));
    SHA256_Final(hash, &sha256);
    for(int i = 0; i < SHA256_DIGEST_LENGTH; i++)
    {
        sprintf(buf, sizeof(buf), "%02x", hash[i]);
        strcat(result, buf);
    }
}

void hmac_sha256(const char* key, int key_len,
const char* input, int input_len,
unsigned char* output, unsigned int* output_len)
{
    HMAC_CTX *h;
    #if OPENSSL_VERSION_NUMBER < 0x10100000L
```

```
HMAC_CTX hmac;
HMAC_CTX_init(&hmac);
h = &hmac;
#else
h = HMAC_CTX_new();
#endif

HMAC_Init_ex(h, key, key_len, EVP_sha256(), NULL);
HMAC_Update(h, ( unsigned char* )input, input_len);
HMAC_Final(h, output, output_len);

#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX_cleanup(h);
#else
HMAC_CTX_free(h);
#endif

}

void hex_encode(const char* input, int input_len, char* output)
{
    static const char* const lut = "0123456789abcdef";

    char add_out[128] = {0};
    char temp[2] = {0};
    for (size_t i = 0; i < input_len; ++i)
    {
        const unsigned char c = input[i];
        temp[0] = lut[c >> 4];
        strcat(add_out, temp);
        temp[0] = lut[c & 15];
        strcat(add_out, temp);
    }
    strncpy(output, add_out, 128);
}

void lowercase(const char * src, char * dst)
{
    for (int i = 0; src[i]; i++)
    {
        dst[i] = tolower(src[i]);
    }
}

int main()
{
    // 密钥参数
    // 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****
    const char* SECRET_ID = getenv("TENCENTCLOUD_SECRET_ID");
```

```
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
const char* SECRET_KEY = getenv("TENCENTCLOUD_SECRET_KEY");
const char* service = "cvm";
const char* host = "cvm.tencentcloudapi.com";
const char* region = "ap-guangzhou";
const char* action = "DescribeInstances";
const char* version = "2017-03-12";
int64_t timestamp = 1551113065;
char date[20] = {0};
get_utc_date(timestamp, date, sizeof(date));

// ***** 步骤 1: 拼接规范请求串 *****
const char* http_request_method = "POST";
const char* canonical_uri = "/";
const char* canonical_query_string = "";
char canonical_headers[100] = {"content-type:application/json; charset=utf-8\nhost:"};
strcat(canonical_headers, host);
strcat(canonical_headers, "\nx-tc-action:");
char value[100] = {0};
lowercase(action, value);
strcat(canonical_headers, value);
strcat(canonical_headers, "\n");
const char* signed_headers = "content-type;host;x-tc-action";
const char* payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name\"}] }";
char hashed_request_payload[100] = {0};
sha256_hex(payload, hashed_request_payload);

char canonical_request[256] = {0};
sprintf(canonical_request, "%s\n%s\n%s\n%s\n%s\n%s", http_request_method,
canonical_uri, canonical_query_string, canonical_headers,
signed_headers, hashed_request_payload);
printf("%s\n", canonical_request);

// ***** 步骤 2: 拼接待签名字符串 *****
const char* algorithm = "TC3-HMAC-SHA256";
char request_timestamp[16] = {0};
sprintf(request_timestamp, "%d", timestamp);
char credential_scope[64] = {0};
strcat(credential_scope, date);
sprintf(credential_scope, "%s/%s/tc3_request", date, service);
char hashed_canonical_request[100] = {0};
sha256_hex(canonical_request, hashed_canonical_request);
char string_to_sign[256] = {0};
sprintf(string_to_sign, "%s\n%s\n%s\n%s", algorithm, request_timestamp,
credential_scope, hashed_canonical_request);
printf("%s\n", string_to_sign);

// ***** 步骤 3: 计算签名 *****
```

```
char k_key[64] = {0};
sprintf(k_key, "%s", "TC3", SECRET_KEY);
unsigned char k_date[64] = {0};
unsigned int output_len = 0;
hmac_sha256(k_key, strlen(k_key), date, strlen(date), k_date, &output_len);
unsigned char k_service[64] = {0};
hmac_sha256(k_date, output_len, service, strlen(service), k_service, &output_len);
unsigned char k_signing[64] = {0};
hmac_sha256(k_service, output_len, "tc3_request", strlen("tc3_request"), k_signing, &output_len);
unsigned char k_hmac_sha_sign[64] = {0};
hmac_sha256(k_signing, output_len, string_to_sign, strlen(string_to_sign), k_hmac_sha_sign, &output_len);

char signature[128] = {0};
hex_encode(k_hmac_sha_sign, output_len, signature);
printf("%s\n", signature);

// ***** 步骤 4: 拼接 Authorization *****
char authorization[512] = {0};
sprintf(authorization, "%s Credential=%s/%s, SignedHeaders=%s, Signature=%s",
algorithm, SECRET_ID, credential_scope, signed_headers, signature);
printf("%s\n", authorization);

char curlcmd[10240] = {0};
sprintf(curlcmd, "curl -X POST https://%s\n \
-H \"Authorization: %s\"\n \
-H \"Content-Type: application/json; charset=utf-8\"\n \
-H \"Host: %s\"\n \
-H \"X-TC-Action: %s\"\n \
-H \"X-TC-Timestamp: %s\"\n \
-H \"X-TC-Version: %s\"\n \
-H \"X-TC-Region: %s\"\n \
-d '%s'\",
host, authorization, host, action, request_timestamp, version, region, payload);
printf("%s\n", curlcmd);
return 0;
}
```

其他语言

- Lua: [GitHub](#)
- Swift: [GitHub](#)
- Dart: [GitHub](#)
- Shell(Bash): [GitHub](#)

签名失败

存在以下签名失败的错误码，请根据实际情况处理。

错误码	错误描述
AuthFailure.SignatureExpire	签名过期。Timestamp 与服务器接收到请求的时间相差不得超过五分钟。

错误码	错误描述
AuthFailure.SecretIdNotFound	密钥不存在。请到控制台查看密钥是否被禁用，是否少复制了字符或者多了字符。
AuthFailure.SignatureFailure	签名错误。可能是签名计算错误，或者签名与实际发送的内容不符合，也有可能是密钥 SecretKey 错误导致的。
AuthFailure.TokenFailure	临时证书 Token 错误。
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）。

签名方法

最近更新时间：2024-12-25 01:59:16

签名方法 v1 简单易用，但是功能和安全性都不如签名方法 v3，推荐使用签名方法 v3。

首次接触，建议使用 [API Explorer](#) 中的“签名串生成”功能，选择签名版本为“API 3.0 签名 v1”，可以生成签名过程进行验证，并提供了部分编程语言的签名示例，也可直接生成 SDK 代码。推荐使用腾讯云 API 配套的 8 种常见的编程语言 SDK，已经封装了签名和请求过程，均已开源，支持 [Python](#)、[Java](#)、[PHP](#)、[Go](#)、[NodeJS](#)、[.NET](#)、[C++](#)、[Ruby](#)。

推荐使用 API Explorer

<> 点击调试

您可以通过 API Explorer 的【签名串生成】模块查看每个接口签名的生成过程。

腾讯云 API 会对每个访问请求进行身份验证，即每个请求都需要在公共请求参数中包含签名信息（Signature）以验证请求者身份。签名信息由安全凭证生成，安全凭证包括 SecretId 和 SecretKey；若用户还没有安全凭证，请前往 [云API密钥页面](#) 申请，否则无法调用云 API 接口。

1. 申请安全凭证

在第一次使用云 API 之前，请前往 [云 API 密钥页面](#) 申请安全凭证。

安全凭证包括 SecretId 和 SecretKey：

- SecretId 用于标识 API 调用者身份
- SecretKey 用于加密签名字符串和服务器端验证签名字符串的密钥。
- 用户必须严格保管安全凭证，避免泄露。

申请安全凭证的具体步骤如下：

1. 登录 [腾讯云管理中心控制台](#)。
2. 前往 [云 API 密钥](#) 的控制台页面
3. 在 [云 API 密钥](#) 页面，单击【新建密钥】即可以创建一对 SecretId/SecretKey。

注意：每个账号最多可以拥有两对 SecretId/SecretKey。

2. 生成签名串

有了安全凭证 SecretId 和 SecretKey 后，就可以生成签名串了。以下是使用签名方法 v1 生成签名串的详细过程：

假设用户的 SecretId 和 SecretKey 分别是：

- SecretId: AKID*****
- SecretKey: *****

注意：这里只是示例，请根据用户实际申请的 SecretId 和 SecretKey 进行后续操作！

以云服务器查看实例列表（DescribeInstances）请求为例，当用户调用这一接口时，其请求参数可能如下：

参数名称	中文	参数值
Action	方法名	DescribeInstances
SecretId	密钥 ID	AKID*****
Timestamp	当前时间戳	1465185768
Nonce	随机正整数	11886

参数名称	中文	参数值
Region	实例所在区域	ap-guangzhou
InstanceIds.0	待查询的实例 ID	ins-09dx96dg
Offset	偏移量	0
Limit	最大允许输出	20
Version	接口版本号	2017-03-12

这里只展示了部分公共参数和接口输入参数，用户可以根据实际需要添加其他参数，例如 Language 和 Token 公共参数。

2.1. 对参数排序

首先对所有请求参数按参数名的字典序（ASCII 码）升序排序。注意：1）只按参数名进行排序，参数值保持对应即可，不参与比大小；2）按 ASCII 码比大小，如 InstanceIds.2 要排在 InstanceIds.12 后面，不是按字母表，也不是按数值。用户可以借助编程语言中的相关排序函数来实现这一功能，如 PHP 中的 ksort 函数。上述示例参数的排序结果如下：

```
{
  'Action' : 'DescribeInstances',
  'InstanceIds.0' : 'ins-09dx96dg',
  'Limit' : 20,
  'Nonce' : 11886,
  'Offset' : 0,
  'Region' : 'ap-guangzhou',
  'SecretId' : 'AKID*****',
  'Timestamp' : 1465185768,
  'Version' : '2017-03-12',
}
```

使用其它程序设计语言开发时，可对上面示例中的参数进行排序，得到的结果一致即可。

2.2. 拼接请求字符串

此步骤生成请求字符串。

将把上一步排序好的请求参数格式化成“参数名称=参数值”的形式，如对 Action 参数，其参数名称为 "Action"，参数值为 "DescribeInstances"，因此格式化后即为 Action=DescribeInstances。

注意：“参数值”为原始值而非 url 编码后的值。

然后将格式化后的各个参数用"&"拼接在一起，最终生成的请求字符串为：

```
Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKID*****&Timestamp=1465185768&Version=2017-03-12
```

2.3. 拼接签名原文字符串

此步骤生成签名原文字符串。

签名原文字符串由以下几个参数构成：

1. 请求方法: 支持 POST 和 GET 方式，这里使用 GET 请求，注意方法为全大写。
2. 请求主机: 查看实例列表(DescribeInstances)的请求域名为: cvm.tencentcloudapi.com。实际的请求域名根据接口所属模块的不同而不同，详见各接口说明。
3. 请求路径: 当前版本云API的请求路径固定为 /。

4. 请求字符串: 即上一步生成的请求字符串。

签名原文串的拼接规则为: 请求方法 + 请求主机 + 请求路径 + ? + 请求字符串。

示例的拼接结果为:

```
GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKID*****&Timestamp=1465185768&Version=2017-03-12
```

2.4. 生成签名串

此步骤生成签名串。

首先使用 HMAC-SHA1 算法对上一步中获得的签名原文字符串进行签名, 然后将生成的签名串使用 Base64 进行编码, 即可获得最终的签名串。

具体代码如下, 以 PHP 语言为例:

```
$secretKey = '*****';
$srcStr = 'GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKID*****&Timestamp=1465185768&Version=2017-03-12';
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));
echo $signStr;
```

最终得到的签名串为:

```
7RAM2xfNMO9EiVTNmPg06MRnCvQ=
```

使用其它程序设计语言开发时, 可用上面示例中的原文进行签名验证, 得到的签名串与例子中的一致即可。

3. 签名串编码

生成的签名串并不能直接作为请求参数, 需要对其进行 URL 编码。

如上一步生成的签名串为 7RAM2xfNMO9EiVTNmPg06MRnCvQ=, 最终得到的签名串请求参数 (Signature) 为: 7RAM2xfNMO9EiVTNmPg06MRnCvQ%3D, 它将用于生成最终的请求 URL。

注意: 如果用户的请求方法是 GET, 或者请求方法为 POST 同时 Content-Type 为 application/x-www-form-urlencoded, 则发送请求时所有请求参数的值均需要做 URL 编码, 参数键和=符号不需要编码。非 ASCII 字符在 URL 编码前需要先用 UTF-8 进行编码。

注意: 有些编程语言的库会自动为所有参数进行 urlencode, 在这种情况下, 就不需要对签名串进行 URL 编码了, 否则两次 URL 编码会导致签名失败。

注意: 其他参数值也需要进行编码, 编码采用 RFC 3986。使用 %XY 对特殊字符例如汉字进行百分比编码, 其中 “X” 和 “Y” 为十六进制字符 (0-9 和大写字母 A-F), 使用小写将引发错误。

4. 签名失败

根据实际情况, 存在以下签名失败的错误码, 请根据实际情况处理。

错误代码	错误描述
AuthFailure.SignatureExpire	签名过期

错误代码	错误描述
AuthFailure.SecretIdNotFound	密钥不存在
AuthFailure.SignatureFailure	签名错误
AuthFailure.TokenFailure	token 错误
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）

5. 签名演示

在实际调用 API 3.0 时，推荐使用配套的腾讯云 SDK 3.0，SDK 封装了签名的过程，开发时只关注产品提供的具体接口即可。详细信息参见 [SDK 中心](#)。当前支持的编程语言有：

- [Python](#)
- [Java](#)
- [PHP](#)
- [Go](#)
- [NodeJS](#)
- [.NET](#)
- [C++](#)
- [Ruby](#)

下面提供了不同产品的生成签名 demo，您可以找到对应的产品参考签名的生成：

- [Signature Demo](#)

为了更清楚的解释签名过程，下面以实际编程语言为例，将上述的签名过程具体实现。请求的域名、调用的接口和参数的取值都以上述签名过程为准，代码只为解释签名过程，并不具备通用性，实际开发请尽量使用 SDK。

最终输出的 url 可能为：`https://cvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKID*****&Signature=7RAM2xfNM09EiVTNmPg06MRnCvQ%3D&Timestamp=1465185768&Version=2017-03-12。`

注意：由于示例中的密钥是虚构的，时间戳也不是系统当前时间，因此如果将此 url 在浏览器中打开或者用 curl 等命令调用时会返回鉴权错误：签名过期。为了得到一个可以正常返回的 url，需要修改示例中的 SecretId 和 SecretKey 为真实的密钥，并使用系统当前时间戳作为 Timestamp。

注意：在下面的示例中，不同编程语言，甚至同一语言每次执行得到的 url 可能都有所不同，表现为参数的顺序不同，但这并不影响正确性。只要所有参数都在，且签名计算正确即可。

注意：以下代码仅适用于 API 3.0，不能直接用于其他的签名流程，请以对应的实际文档为准。

Java

```
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.Random;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;

public class TencentCloudAPIDemo {
```

```
private final static String CHARSET = "UTF-8";

public static String sign(String s, String key, String method) throws Exception {
    Mac mac = Mac.getInstance(method);
    SecretKeySpec secretKeySpec = new SecretKeySpec(key.getBytes(CHARSET), mac.getAlgorithm());
    mac.init(secretKeySpec);
    byte[] hash = mac.doFinal(s.getBytes(CHARSET));
    return DatatypeConverter.printBase64Binary(hash);
}

public static String getStringToSign(TreeMap<String, Object> params) {
    StringBuilder s2s = new StringBuilder("GETcvm.tencentcloudapi.com/?");
    // 签名时要求对参数进行字典排序, 此处用TreeMap保证顺序
    for (String k : params.keySet()) {
        s2s.append(k).append("=").append(params.get(k).toString()).append("&");
    }
    return s2s.toString().substring(0, s2s.length() - 1);
}

public static String getUrl(TreeMap<String, Object> params) throws UnsupportedEncodingException {
    StringBuilder url = new StringBuilder("https://cvm.tencentcloudapi.com/?");
    // 实际请求的url中对参数顺序没有要求
    for (String k : params.keySet()) {
        // 需要对请求串进行urlencode, 由于key都是英文字母, 故此处仅对其value进行urlencode
        url.append(k).append("=").append(URLEncoder.encode(params.get(k).toString(), CHARSET)).append("&");
    }
    return url.toString().substring(0, url.length() - 1);
}

public static void main(String[] args) throws Exception {
    TreeMap<String, Object> params = new TreeMap<String, Object>(); // TreeMap可以自动排序
    // 实际调用时应当使用随机数, 例如: params.put("Nonce", new Random().nextInt(java.lang.Integer.MAX_VALUE));
    params.put("Nonce", 11886); // 公共参数
    // 实际调用时应当使用系统当前时间, 例如: params.put("Timestamp", System.currentTimeMillis() / 1000);
    params.put("Timestamp", 1465185768); // 公共参数
    // 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****
    params.put("SecretId", System.getenv("TENCENTCLOUD_SECRET_ID")); // 公共参数
    params.put("Action", "DescribeInstances"); // 公共参数
    params.put("Version", "2017-03-12"); // 公共参数
    params.put("Region", "ap-guangzhou"); // 公共参数
    params.put("Limit", 20); // 业务参数
    params.put("Offset", 0); // 业务参数
    params.put("InstanceIds.0", "ins-09dx96dg"); // 业务参数
    // 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
    params.put("Signature", sign(getStringToSign(params), System.getenv("TENCENTCLOUD_SECRET_KEY"), "HmacSHA1")); // 公共参数
    System.out.println(getUrl(params));
}
}
```

Python

注意：如果是在 Python 2 环境中运行，需要先安装 requests 依赖包： `pip install requests` 。

```
# -*- coding: utf8 -*-
import base64
import hashlib
import hmac
import os
import time

import requests

# 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****
secret_id = os.environ.get("TENCENTCLOUD_SECRET_ID")
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
secret_key = os.environ.get("TENCENTCLOUD_SECRET_KEY")

def get_string_to_sign(method, endpoint, params):
    s = method + endpoint + "/"
    query_str = "&".join("%s=%s" % (k, params[k]) for k in sorted(params))
    return s + query_str

def sign_str(key, s, method):
    hmac_str = hmac.new(key.encode("utf8"), s.encode("utf8"), method).digest()
    return base64.b64encode(hmac_str)

if __name__ == '__main__':
    endpoint = "cvm.tencentcloudapi.com"
    data = {
        'Action': 'DescribeInstances',
        'InstanceIds.0': 'ins-09dx96dg',
        'Limit': 20,
        'Nonce': 11886,
        'Offset': 0,
        'Region': 'ap-guangzhou',
        'SecretId': secret_id,
        'Timestamp': 1465185768, # int(time.time())
        'Version': '2017-03-12'
    }
    s = get_string_to_sign("GET", endpoint, data)
    data["Signature"] = sign_str(secret_key, s, hashlib.sha1)
    print(data["Signature"])
    # 此处会实际调用，成功后可能产生计费
    # resp = requests.get("https://" + endpoint, params=data)
    # print(resp.url)
```

Golang

```
package main

import (
    "bytes"
    "crypto/hmac"
    "crypto/sha1"
    "encoding/base64"
    "fmt"
    "os"
    "sort"
    "strconv"
)

func main() {
    // 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****
    secretId := os.Getenv("TENCENTCLOUD_SECRET_ID")
    // 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
    secretKey := os.Getenv("TENCENTCLOUD_SECRET_KEY")
    params := map[string]string{
        "Nonce": "11886",
        "Timestamp": strconv.Itoa(1465185768),
        "Region": "ap-guangzhou",
        "SecretId": secretId,
        "Version": "2017-03-12",
        "Action": "DescribeInstances",
        "InstanceIds.0": "ins-09dx96dg",
        "Limit": strconv.Itoa(20),
        "Offset": strconv.Itoa(0),
    }

    var buf bytes.Buffer
    buf.WriteString("GET")
    buf.WriteString("cvm.tencentcloudapi.com")
    buf.WriteString("/")
    buf.WriteString("?")

    // sort keys by ascii asc order
    keys := make([]string, 0, len(params))
    for k, _ := range params {
        keys = append(keys, k)
    }
    sort.Strings(keys)

    for i := range keys {
        k := keys[i]
        buf.WriteString(k)
        buf.WriteString("=")
        buf.WriteString(params[k])
        buf.WriteString("&")
    }
}
```

```
}

buf.Truncate(buf.Len() - 1)

hashed := hmac.New(sha1.New, []byte(secretKey))
hashed.Write(buf.Bytes())

fmt.Println(base64.StdEncoding.EncodeToString(hashed.Sum(nil)))
}
```

PHP

```
<?php
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****
$secretId = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
$secretKey = getenv("TENCENTCLOUD_SECRET_KEY");
$params["Nonce"] = 11886;//rand();
$params["Timestamp"] = 1465185768;//time();
$params["Region"] = "ap-guangzhou";
$params["SecretId"] = $secretId;
$params["Version"] = "2017-03-12";
$params["Action"] = "DescribeInstances";
$params["InstanceIds.0"] = "ins-09dx96dg";
$params["Limit"] = 20;
$params["Offset"] = 0;

ksort($params);

$signStr = "GETcvm.tencentcloudapi.com/?";
foreach ( $params as $key => $value ) {
    $signStr = $signStr . $key . "=" . $value . "&";
}
$signStr = substr($signStr, 0, -1);

$signature = base64_encode(hash_hmac("sha1", $signStr, $secretKey, true));
echo $signature.PHP_EOL;
// need to install and enable curl extension in php.ini
// $params["Signature"] = $signature;
// $url = "https://cvm.tencentcloudapi.com/?".http_build_query($params);
// echo $url.PHP_EOL;
// $ch = curl_init();
// curl_setopt($ch, CURLOPT_URL, $url);
// $output = curl_exec($ch);
// curl_close($ch);
// echo json_decode($output);
```

Ruby

```
# -*- coding: UTF-8 -*-
# require ruby>=2.3.0
require 'time'
require 'openssl'
require 'base64'

# 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****
secret_id = ENV["TENCENTCLOUD_SECRET_ID"]
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
secret_key = ENV["TENCENTCLOUD_SECRET_KEY"]

method = 'GET'
endpoint = 'cvm.tencentcloudapi.com'
data = {
  'Action' => 'DescribeInstances',
  'InstanceIds.0' => 'ins-09dx96dg',
  'Limit' => 20,
  'Nonce' => 11886,
  'Offset' => 0,
  'Region' => 'ap-guangzhou',
  'SecretId' => secret_id,
  'Timestamp' => 1465185768, # Time.now.to_i
  'Version' => '2017-03-12',
}
sign = method + endpoint + '/'
params = []
data.sort.each do |item|
  params << "#{item[0]}=#{item[1]}"
end
sign += params.join('&')
digest = OpenSSL::Digest.new('sha1')
data['Signature'] = Base64.encode64(OpenSSL::HMAC.digest(digest, secret_key, sign))
puts data['Signature']

# require 'net/http'
# uri = URI('https://' + endpoint)
# uri.query = URI.encode_www_form(data)
# p uri
# res = Net::HTTP.get_response(uri)
# puts res.body
```

DotNet

```
using System;
using System.Collections.Generic;
using System.Net;
using System.Security.Cryptography;
using System.Text;
```

```
public class Application {
    public static string Sign(string signKey, string secret)
    {
        string signRet = string.Empty;
        using (HMACSHA1 mac = new HMACSHA1(Encoding.UTF8.GetBytes(signKey)))
        {
            byte[] hash = mac.ComputeHash(Encoding.UTF8.GetBytes(secret));
            signRet = Convert.ToBase64String(hash);
        }
        return signRet;
    }

    public static string MakeSignPlainText(SortedDictionary<string, string> requestParams, string requestMethod, string requestHost, string requestPath)
    {
        string retStr = "";
        retStr += requestMethod;
        retStr += requestHost;
        retStr += requestPath;
        retStr += "?";
        string v = "&";
        foreach (string key in requestParams.Keys)
        {
            v += string.Format("{0}={1}&", key, requestParams[key]);
        }
        retStr += v.TrimEnd('&');
        return retStr;
    }

    public static void Main(string[] args)
    {
        // 密钥参数
        // 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****
        string SECRET_ID = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_ID");
        // 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
        string SECRET_KEY = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_KEY");

        string endpoint = "cvm.tencentcloudapi.com";
        string region = "ap-guangzhou";
        string action = "DescribeInstances";
        string version = "2017-03-12";
        double RequestTimestamp = 1465185768; // 时间戳 2019-02-26 00:44:25, 此参数作为示例, 以实际为准
        // long timestamp = ToTimestamp() / 1000;
        // string requestTimestamp = timestamp.ToString();
        Dictionary<string, string> param = new Dictionary<string, string>();
        param.Add("Limit", "20");
        param.Add("Offset", "0");
        param.Add("InstanceIds.0", "ins-09dx96dg");
        param.Add("Action", action);
        param.Add("Nonce", "11886");
        // param.Add("Nonce", Math.Abs(new Random().Next()).ToString());
    }
}
```

```
param.Add("Timestamp", RequestTimestamp.ToString());
param.Add("Version", version);

param.Add("SecretId", SECRET_ID);
param.Add("Region", region);
SortedDictionary<string, string> headers = new SortedDictionary<string, string>(param, StringComparer.Ordinal);
string sigInParameter = MakeSignPlainText(headers, "GET", endpoint, "/");
string sigOutParam = Sign(SECRET_KEY, sigInParameter);
Console.WriteLine(sigOutParam);
}
}
```

NodeJS

```
const crypto = require('crypto');

function get_req_url(params, endpoint){
    params['Signature'] = encodeURIComponent(params['Signature']);
    const url_strParam = sort_params(params)
    return "https://" + endpoint + "/" + url_strParam.slice(1);
}

function formatSignString(reqMethod, endpoint, path, strParam){
    let strSign = reqMethod + endpoint + path + "?" + strParam.slice(1);
    return strSign;
}

function sha1(secretKey, strsign){
    let signMethodMap = {'HmacSHA1': "sha1"};
    let hmac = crypto.createHmac(signMethodMap['HmacSHA1'], secretKey || "");
    return hmac.update(Buffer.from(strsign, 'utf8')).digest('base64')
}

function sort_params(params){
    let strParam = "";
    let keys = Object.keys(params);
    keys.sort();
    for (let k in keys) {
        //k = k.replace(/_/g, '.');
        strParam += ("&" + keys[k] + "=" + params[keys[k]]);
    }
    return strParam
}

function main(){
    // 密钥参数
    // 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****
    const SECRET_ID = process.env.TENCENTCLOUD_SECRET_ID
```

```
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
const SECRET_KEY = process.env.TENCENTCLOUD_SECRET_KEY

const endpoint = "cvm.tencentcloudapi.com"
const Region = "ap-guangzhou"
const Version = "2017-03-12"
const Action = "DescribeInstances"
const Timestamp = 1465185768 // 时间戳 2016-06-06 12:02:48, 此参数作为示例, 以实际为准
// const Timestamp = Math.round(Date.now() / 1000)
const Nonce = 11886 // 随机正整数
//const nonce = Math.round(Math.random() * 65535)

let params = {};
params['Action'] = Action;
params['InstanceIds.0'] = 'ins-09dx96dg';
params['Limit'] = 20;
params['Offset'] = 0;
params['Nonce'] = Nonce;
params['Region'] = Region;
params['SecretId'] = SECRET_ID;
params['Timestamp'] = Timestamp;
params['Version'] = Version;

// 1. 对参数排序, 并拼接请求字符串
strParam = sort_params(params)

// 2. 拼接签名原字符串
const reqMethod = "GET";
const path = "/";
strSign = formatSignString(reqMethod, endpoint, path, strParam)
// console.log(strSign)

// 3. 生成签名串
params['Signature'] = sha1(SECRET_KEY, strSign)
console.log(params['Signature'])

// 4. 进行url编码并拼接请求url
// const req_url = get_req_url(params, endpoint)
// console.log(params['Signature'])
// console.log(req_url)
}

main()
```

返回结果

最近更新时间：2024-03-12 01:48:41

云 API 3.0 接口默认返回 JSON 数据，返回非 JSON 格式的接口会在文档中做出说明。返回 JSON 数据时最大限制为 50 MB，如果返回的数据超过最大限制，请求会失败并返回内部错误。请根据接口文档中给出的过滤功能（例如时间范围）或者分页功能，控制返回数据不要过大。

注意：目前只要请求被服务端正常处理了，响应的 HTTP 状态码均为 200。例如返回的消息体里的错误码是签名失败，但 HTTP 状态码是 200，而不是 401。

正确返回结果

以云服务器的接口查看实例状态列表 (DescribeInstancesStatus) 2017-03-12 版本为例，若调用成功，其可能的返回如下为：

```
{
  "Response": {
    "TotalCount": 0,
    "InstanceStatusSet": [],
    "RequestId": "b5b41468-520d-4192-b42f-595cc34b6c1c"
  }
}
```

- Response 及其内部的 RequestId 是固定的字段，无论请求成功与否，只要 API 处理了，则必定会返回。
- RequestId 用于一个 API 请求的唯一标识，如果 API 出现异常，可以联系 [腾讯云客服](#) 或 [提交工单](#)，并提供该 ID 来解决问题。
- 除了固定的字段外，其余均为具体接口定义的字段，不同的接口所返回的字段参见接口文档中的定义。此例中的 TotalCount 和 InstanceStatusSet 均为 DescribeInstancesStatus 接口定义的字段，由于调用请求的用户暂时还没有云服务器实例，因此 TotalCount 在此情况下的返回值为 0，InstanceStatusSet 列表为空。

错误返回结果

若调用失败，其返回值示例如下为：

```
{
  "Response": {
    "Error": {
      "Code": "AuthFailure.SignatureFailure",
      "Message": "The provided credentials could not be validated. Please check your signature is correct."
    },
    "RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
  }
}
```

- Error 的出现代表着该请求调用失败。Error 字段连同其内部的 Code 和 Message 字段在调用失败时是必定返回的。
- Code 表示具体出错的错误码，当请求出错时可以先根据该错误码在公共错误码和当前接口对应的错误码列表里面查找对应原因和解决方案。
- Message 显示出了这个错误发生的具体原因，随着业务发展或体验优化，此文本可能会经常保持变更或更新，用户不应依赖这个返回值。
- RequestId 用于一个 API 请求的唯一标识，如果 API 出现异常，可以联系 [腾讯云客服](#) 或 [提交工单](#)，并提供该 ID 来解决问题。

公共错误码

返回结果中如果存在 Error 字段，则表示调用 API 接口失败。Error 中的 Code 字段表示错误码，所有业务都可能出现的错误码为公共错误码。完整的错误码列表请参考本产品“API 文档”目录下的“错误码”页面。

参数类型

最近更新时间：2022-08-10 06:47:30

目前腾讯云 API 3.0 输入参数和输出参数支持如下几种数据格式：

- String: 字符串。
- Integer: 整型，上限为无符号64位整数。SDK 3.0 不同编程语言支持的类型有所差异，建议以所使用编程语言的最大整型定义，例如 Go 的 `uint64`。
- Boolean: 布尔型。
- Float: 浮点型。
- Double: 双精度浮点型。
- Date: 字符串，日期格式。例如：2022-01-01。
- Timestamp: 字符串，时间格式。例如：2022-01-01 00:00:00。
- Timestamp ISO8601: ISO 8601 是由国际标准化组织（International Organization for Standardization，ISO）发布的关于日期和时间格式的国际标准，对应国标《GB/T 7408-2005 数据元和交换格式信息交换 日期和时间表示法》。建议以所使用编程语言的标准库进行格式解析。例如：2022-01-01T00:00:00+08:00。
- Binary: 二进制内容，需要以特定协议请求和解析。

写接口

批量创建标签

最近更新时间：2025-04-25 01:54:31

1. 接口描述

接口请求域名：tag.tencentcloudapi.com。

本接口用于创建多对标签键和标签值

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

<> 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
Action	是	String	公共参数 ，本接口取值：CreateTags。
Version	是	String	公共参数 ，本接口取值：2018-08-13。
Region	否	String	公共参数 ，本接口不需要传递此参数。
Tags.N	否	Array of Tag	标签列表。 N取值范围：0~9

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 创建标签

输入示例

```
POST / HTTP/1.1
Host: tag.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: CreateTags
<公共请求参数>

{
```

```
"Tags": [  
  {  
    "TagKey": "09221",  
    "TagValue": "092211"  
  },  
  {  
    "TagKey": "09221",  
    "TagValue": "092212"  
  }  
]
```

输出示例

```
{  
  "Response": {  
    "RequestId": "eb60900d-8e00-4bfd-bb92-64a*****"  
  }  
}
```

5. 开发者资源

腾讯云 API 平台

[腾讯云 API 平台](#) 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 [API Inspector](#) 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 [API Explorer](#) 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Java: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for PHP: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Go: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Node.js: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for .NET: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for C++: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Ruby: [GitHub](#), [Gitee](#)

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

错误码	描述
FailedOperation	操作失败。
InvalidParameter	参数错误。
InvalidParameter.ReservedTagKey	系统预留标签键 qcloud、tencent和project 禁止创建。
InvalidParameter.Tag	Tag参数错误。
InvalidParameterValue.ReservedTagKey	系统预留标签键 qcloud、tencent和project 禁止创建。
InvalidParameterValue.TagKeyCharacterIllegal	标签键包含非法字符。
InvalidParameterValue.TagKeyEmpty	标签键不能为空值。
InvalidParameterValue.TagKeyLengthExceeded	标签键长度超过限制。
InvalidParameterValue.TagValueCharacterIllegal	标签值包含非法字符。
InvalidParameterValue.TagValueEmpty	标签值不能为空值。
InvalidParameterValue.TagValueLengthExceeded	标签值长度超过限制。
LimitExceeded.TagKey	用户创建标签键达到上限数 1000。
LimitExceeded.TagValue	单个标签键对应标签值达到上限数 1000。
ResourceInUse.TagDuplicate	标签已存在。

批量删除标签

最近更新时间：2025-04-25 01:54:31

1. 接口描述

接口请求域名：tag.tencentcloudapi.com。

本接口用于批量删除标签键和标签值。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

</> 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
Action	是	String	公共参数 ，本接口取值：DeleteTags。
Version	是	String	公共参数 ，本接口取值：2018-08-13。
Region	否	String	公共参数 ，本接口不需要传递此参数。
Tags.N	是	Array of Tag	标签列表。 N取值范围：0~9

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 删除标签

输入示例

```
POST / HTTP/1.1
Host: tag.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DeleteTags
<公共请求参数>

{
  "Tags": [
    {
```

```
"TagKey": "09221",
"TagValue": "092211"
},
{
"TagKey": "09221",
"TagValue": "092212"
}
]
```

输出示例

```
{
"Response": {
"RequestId": "131fd253-b2de-4190-a1c1-13ce*****"
}
}
```

5. 开发者资源

腾讯云 API 平台

[腾讯云 API 平台](#) 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 [API Inspector](#) 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 [API Explorer](#) 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Java: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for PHP: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Go: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Node.js: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for .NET: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for C++: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Ruby: [GitHub](#), [Gitee](#)

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

错误码	描述
FailedOperation	操作失败。

错误码	描述
FailedOperation.TagAttachedQuota	标签已经关联配额。
FailedOperation.TagAttachedResource	已关联资源的标签无法删除。
InvalidParameter	参数错误。
InvalidParameter.ReservedTagKey	系统预留标签键 qcloud、tencent和project 禁止创建。
InvalidParameterValue.TagKeyCharacterIllegal	标签键包含非法字符。
InvalidParameterValue.TagKeyEmpty	标签键不能为空值。
InvalidParameterValue.TagKeyLengthExceeded	标签键长度超过限制。
InvalidParameterValue.TagValueCharacterIllegal	标签值包含非法字符。
InvalidParameterValue.TagValueEmpty	标签值不能为空值。
InvalidParameterValue.TagValueLengthExceeded	标签值长度超过限制。
ResourceNotFound.TagNotExist	标签不存在。

为资源解绑标签

最近更新时间：2025-04-25 01:54:30

1. 接口描述

接口请求域名：tag.tencentcloudapi.com。

指定的多个云产品的多个云资源统一解绑标签。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

</> 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
Action	是	String	公共参数 ，本接口取值：UntagResources。
Version	是	String	公共参数 ，本接口取值：2018-08-13。
Region	否	String	公共参数 ，本接口不需要传递此参数。
ResourceList.N	是	Array of String	资源六段式列表。腾讯云使用资源六段式描述一个资源。可参考 访问管理-概览-接口列表-资源六段式信息 例如：ResourceList.1 = qcs::\${ServiceType}:\${Region}:uin/\${Account}:\${ResourcePrefix}/\${ResourceId}。 N取值范围：0~9 示例值：["qcs::cvm:ap-beijing:uin/1234567:instance/ins-123"]
TagKeys.N	是	Array of String	标签键。 取值范围：0~9 示例值：["key1"]

3. 输出参数

参数名称	类型	描述
FailedResources	Array of FailedResource	失败资源信息。 解绑标签成功时，返回的FailedResources为空。 解绑标签失败或部分失败时，返回的FailedResources会显示失败资源的详细信息。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 为资源解绑标签

输入示例

```
POST / HTTP/1.1
Host: tag.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: UnTagResources
<公共请求参数>

{
  "ResourceList": [
    "qcs::cvm:ap-beijing:uin/10000055****:instance/****",
    "qcs::cvm:ap-beijing:uin/10000055****:instance/****"
  ],
  "TagKeys": [
    "RWFGaltFlR",
    "LbDKGjeagV"
  ]
}
```

输出示例

```
{
  "Response": {
    "FailedResources": [],
    "RequestId": "cbc7b26c-3fff-4a3c-82f0-c6ebc2ff31e3"
  }
}
```

5. 开发者资源

腾讯云 API 平台

[腾讯云 API 平台](#) 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 [API Inspector](#) 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 [API Explorer](#) 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Java: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for PHP: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Go: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Node.js: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for .NET: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for C++: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Ruby: [GitHub](#), [Gitee](#)

命令行工具

• [Tencent Cloud CLI 3.0](#)

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

错误码	描述
AuthFailure.UnauthorizedOperation	未通过CAM鉴权。
FailedOperation	操作失败。
FailedOperation.PolicyEnforcedInvalid	标签策略强制打标签限制
FailedOperation.ResourceTagProcessing	资源标签正在处理中。
InvalidParameter	参数错误。
InvalidParameter.ReservedTagKey	系统预留标签键 qcloud、tencent和project 禁止创建。
InvalidParameter.Tag	Tag参数错误。
InvalidParameterValue.ResourceDescriptionError	资源描述错误。
InvalidParameterValue.TagKeyCharacterIllegal	标签键包含非法字符。
InvalidParameterValue.TagKeyDuplicate	TagList中存在重复的TagKey。
InvalidParameterValue.TagKeyEmpty	标签键不能为空值。
InvalidParameterValue.TagKeyLengthExceeded	标签键长度超过限制。
InvalidParameterValue.TagValueCharacterIllegal	标签值包含非法字符。
LimitExceeded.ResourceNumPerRequest	单次请求的资源数达到上限。
LimitExceeded.TagNumPerRequest	单次请求的标签数超过上限。

为资源绑定标签

最近更新时间：2025-04-25 01:54:30

1. 接口描述

接口请求域名：tag.tencentcloudapi.com。

为指定的多个云产品的多个云资源统一创建并绑定标签。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

</> 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
Action	是	String	公共参数 ，本接口取值：TagResources。
Version	是	String	公共参数 ，本接口取值：2018-08-13。
Region	否	String	公共参数 ，本接口不需要传递此参数。
ResourceList.N	是	Array of String	待绑定的云资源，用标准的资源六段式表示。正确的资源六段式请参考： 标准的资源六段式 和 支持标签的云产品及资源描述方式 。 N取值范围：0~9 示例值：["qcs::cvm:ap-beijing:uin/1234567:instance/ins-123"]
Tags.N	是	Array of Tag	标签键和标签值。 如果指定多个标签，则会为指定资源同时创建并绑定该多个标签。 同一个资源上的同一个标签键只能对应一个标签值。如果您尝试添加已有标签键，则对应的标签值会更新为新值。 如果标签不存在会为您自动创建标签。 N取值范围：0~9

3. 输出参数

参数名称	类型	描述
FailedResources	Array of FailedResource	失败资源信息。 创建并绑定标签成功时，返回的FailedResources为空。 创建并绑定标签失败或部分失败时，返回的FailedResources会显示失败资源的详细信息。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 为资源绑定标签

为资源绑定标签

输入示例

```
POST / HTTP/1.1
Host: tag.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: TagResources
<公共请求参数>

{
  "ResourceList": [
    "qcs::cvm:ap-beijing:uin/10000055****:instance/ins-****",
    "qcs::cvm:ap-shanghai:uin/10000055****:volume/****"
  ],
  "Tags": [
    {
      "TagKey": "11",
      "TagValue": "11"
    },
    {
      "TagKey": "22",
      "TagValue": "22"
    }
  ]
}
```

输出示例

```
{
  "Response": {
    "FailedResources": [],
    "RequestId": "a4c9f540-b04e-47c4-a267-4a398b70d2f5"
  }
}
```

5. 开发者资源

腾讯云 API 平台

[腾讯云 API 平台](#) 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 [API Inspector](#) 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 [API Explorer](#) 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Java: [GitHub](#), [Gitee](#)

- Tencent Cloud SDK 3.0 for PHP: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Go: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Node.js: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for .NET: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for C++: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Ruby: [GitHub](#), [Gitee](#)

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

错误码	描述
AuthFailure.UnauthorizedOperation	未通过CAM鉴权。
FailedOperation	操作失败。
FailedOperation.PolicyEnforcedInvalid	标签策略强制打标签限制
FailedOperation.ResourceTagProcessing	资源标签正在处理中。
InvalidParameter	参数错误。
InvalidParameter.ReservedTagKey	系统预留标签键 qcloud、tencent和project 禁止创建。
InvalidParameter.Tag	Tag参数错误。
InvalidParameter.UnsupportedService	当前业务不支持标签操作。
InvalidParameterValue.ResourceDescriptionError	资源描述错误。
InvalidParameterValue.TagKeyCharacterIllegal	标签键包含非法字符。
InvalidParameterValue.TagKeyDuplicate	TagList中存在重复的TagKey。
InvalidParameterValue.TagKeyEmpty	标签键不能为空值。
InvalidParameterValue.TagKeyLengthExceeded	标签键长度超过限制。
InvalidParameterValue.TagValueCharacterIllegal	标签值包含非法字符。
InvalidParameterValue.TagValueEmpty	标签值不能为空值。
InvalidParameterValue.TagValueLengthExceeded	标签值长度超过限制。
LimitExceeded.ResourceNumPerRequest	单次请求的资源数达到上限。
LimitExceeded.TagNumPerRequest	单次请求的标签数超过上限。
LimitExceeded.TagValue	单个标签键对应标签值达到上限数 1000。

读接口

查询资源标签列表

最近更新时间：2025-04-25 01:54:28

1. 接口描述

接口请求域名：tag.tencentcloudapi.com。

查询绑定了标签的资源列表。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

<> 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
Action	是	String	公共参数 ，本接口取值：GetResources。
Version	是	String	公共参数 ，本接口取值：2018-08-13。
Region	否	String	公共参数 ，本接口不需要传递此参数。
ResourceList.N	否	Array of String	资源六段式列表。腾讯云使用资源六段式描述一个资源。 例如：ResourceList.1 = qcs::\${ServiceType}:\${Region}:\${Account}:\${ResourcePreifx}/\${ResourceId}。 如果传入了此参数会返回所有匹配的资源列表，指定的MaxResults会失效。 N取值范围：0~9 示例值：["qcs::cvm:ap-beijing:uin/1234567:instance/ins-123"]
TagFilters.N	否	Array of TagFilter	标签键和标签值。 指定多个标签，会查询同时绑定了该多个标签的资源。 N取值范围：0~5。 每个TagFilters中的TagValue最多支持10个
PaginationToken	否	String	从上一页的响应中获取的下一页的Token值。 如果是第一次请求，设置为空。 示例值：POzXXYGXh2fRDHxx7PGaJnLApcyluAYBC5HX25gKI4Sz2R*
MaxResults	否	Integer	每一页返回的数据最大条数，最大200。 缺省值：50。 示例值：50

3. 输出参数

参数名称	类型	描述
PaginationToken	String	获取的下一页的Token值 示例值：

参数名称	类型	描述
		POzXXYGXh2fRDHxx7PGaJnLApcyluAYBC5HX25gKI4Sz2R*
ResourceTagMappingList	Array of ResourceTagMapping	资源及关联的标签(键和值)列表
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询资源标签列表

输入示例

```
POST / HTTP/1.1
Host: tag.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: GetResources
<公共请求参数>

{
  "ResourceList": [
    "qcs::cvm:ap-beijing:uin/100000558920:instance/ins-123",
    "qcs::cvm:ap-shanghai:uin/100000558920:instance/ins-345"
  ],
  "TagFilters": [
    {
      "TagKey": "11",
      "TagValue": [
        "11"
      ]
    },
    {
      "TagKey": "22",
      "TagValue": [
        "22"
      ]
    }
  ]
}
```

输出示例

```
{
  "Response": {
    "PaginationToken": "",
    "ResourceTagMappingList": [
      {
```

```
"Resource": "qcs::cvm:ap-beijing:uin/100000558920:instance/ins-123",
"Tags": [
  {
    "TagKey": "22",
    "TagValue": "22"
  },
  {
    "TagKey": "11",
    "TagValue": "11"
  }
],
{
  "Resource": "qcs::cvm:ap-shanghai:uin/100000558920:instance/ins-345",
  "Tags": [
    {
      "TagKey": "22",
      "TagValue": "22"
    },
    {
      "TagKey": "11",
      "TagValue": "11"
    }
  ]
},
{
  "RequestId": "07cfd251-4ed1-426c-8133-e04a583063c0"
}
}
```

5. 开发者资源

腾讯云 API 平台

[腾讯云 API 平台](#) 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 [API Inspector](#) 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 [API Explorer](#) 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Java: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for PHP: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Go: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Node.js: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for .NET: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for C++: [GitHub](#), [Gitee](#)

- Tencent Cloud SDK 3.0 for Ruby: [GitHub](#), [Gitee](#)

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

错误码	描述
AuthFailure.UnauthorizedOperation	未通过CAM鉴权。
FailedOperation	操作失败。
FailedOperation.ResourceTagProcessing	资源标签正在处理中。
InvalidParameter	参数错误。
InvalidParameter.PaginationTokenInvalid	PaginationToken参数非法。
InvalidParameter.ReservedTagKey	系统预留标签键 qcloud、tencent和project 禁止创建。
InvalidParameter.Tag	Tag参数错误。
InvalidParameterValue.ResourceDescriptionError	资源描述错误。
InvalidParameterValue.TagKeyDuplicate	TagList中存在重复的TagKey。
InvalidParameterValue.TagKeyEmpty	标签键不能为空值。
InvalidParameterValue.TagKeyLengthExceeded	标签键长度超过限制。
InvalidParameterValue.TagValueCharacterIllegal	标签值包含非法字符。
InvalidParameterValue.TagValueLengthExceeded	标签值长度超过限制。
LimitExceeded.ResourceNumPerRequest	单次请求的资源数达到上限。
LimitExceeded.TagNumPerRequest	单次请求的标签数超过上限。
LimitExceeded.TagValue	单个标签键对应标签值达到上限数 1000。

查询标签键列表

最近更新时间：2025-04-25 01:54:28

1. 接口描述

接口请求域名：tag.tencentcloudapi.com。

查询标签键列表。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

</> 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
Action	是	String	公共参数 ，本接口取值：GetTagKeys。
Version	是	String	公共参数 ，本接口取值：2018-08-13。
Region	否	String	公共参数 ，本接口不需要传递此参数。
PaginationToken	否	String	从上一页的响应中获取的下一页的Token值。 如果是第一次请求，设置为空。 示例值：fgy43h****
MaxResults	否	Integer	每一页返回的数据最大条数，最大1000。 缺省值：50。 示例值：50
Category	否	String	标签类型。取值： Custom：自定义标签。System：系统标签。All：全部标签。默认值：All。 示例值：All

3. 输出参数

参数名称	类型	描述
PaginationToken	String	获取的下一页的Token值，如果当前是最后一页，返回为空 示例值：fgy43h****
TagKeys	Array of String	标签键信息。 示例值：["key1"]
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询标签键列表

输入示例

```
POST / HTTP/1.1
Host: tag.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: GetTagKeys
<公共请求参数>

{
  "PaginationToken": "oaWvQ5SFPGhUtWMA3LKTkgryBtKrDB215AUhaE*****",
  "MaxResults": "10"
}
```

输出示例

```
{
  "Response": {
    "PaginationToken": "kF-AjkgVQoB_9GwaBebsjvXudAjBadBTakA1WqPbwR66KjGE0INYQhQ*****",
    "TagKeys": [
      "testKey1",
      "testKey2"
    ],
    "RequestId": "104eeaa2-c268-43e5-b073-0fade*****"
  }
}
```

5. 开发者资源

腾讯云 API 平台

[腾讯云 API 平台](#) 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 [API Inspector](#) 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 [API Explorer](#) 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Java: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for PHP: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Go: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Node.js: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for .NET: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for C++: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Ruby: [GitHub](#), [Gitee](#)

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

错误码	描述
InvalidParameter	参数错误。
InvalidParameter.PaginationTokenInvalid	PaginationToken参数非法。
InvalidParameterValue.ReservedTagKey	系统预留标签键 qcloud、tencent和project 禁止创建。
InvalidParameterValue.TagKeyLengthExceeded	标签键长度超过限制。
LimitExceeded	超过配额限制。

查询标签值列表

最近更新时间：2025-04-25 01:54:28

1. 接口描述

接口请求域名：tag.tencentcloudapi.com。

用于查询已建立的标签列表中的标签值。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

</> 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
Action	是	String	公共参数 ，本接口取值：GetTagValues。
Version	是	String	公共参数 ，本接口取值：2018-08-13。
Region	否	String	公共参数 ，本接口不需要传递此参数。
TagKeys.N	是	Array of String	标签键。 返回所有标签键列表对应的标签值。 最大长度：20 示例值：["key1"]
PaginationToken	否	String	从上一页的响应中获取的下一页的Token值。 如果是第一次请求，设置为空。 示例值：POzXXYGXh2fRDHxx7PGaJnLApcyluAYBC5HX25gKI4Sz2R*
MaxResults	否	Integer	每一页返回的数据最大条数，最大1000。 缺省值：50。 示例值：50
Category	否	String	标签类型。取值：Custom：自定义标签。System：系统标签。All：全部标签。默认值：All。 示例值：All

3. 输出参数

参数名称	类型	描述
PaginationToken	String	获取的下一页的Token值，如果当前是最后一页，返回为空 示例值：POzXXYGXh2fRDHxx7PGaJnLApcyluAYBC5HX25gKI4Sz2R*
Tags	Array of Tag	标签列表。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询标签值列表

输入示例

```
POST / HTTP/1.1
Host: tag.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: GetTagValues
<公共请求参数>

{
  "PaginationToken": "CvHxL9Up8UzmIeLZKL-pSEjff0aDNyMkn5kyTyn2M*****",
  "TagKeys": [
    "key1"
  ],
  "MaxResults": "1"
}
```

输出示例

```
{
  "Response": {
    "PaginationToken": "j5YP3W_BJau_FcWBKPPHzvDIrG01rHqMuKyxW*****",
    "Tags": [
      {
        "TagKey": "key1",
        "TagValue": "value1"
      }
    ],
    "RequestId": "7337c479-d89b-4c4c-be6d-c7*****"
  }
}
```

5. 开发者资源

腾讯云 API 平台

[腾讯云 API 平台](#) 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 [API Inspector](#) 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 [API Explorer](#) 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Java: [GitHub](#), [Gitee](#)

- Tencent Cloud SDK 3.0 for PHP: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Go: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Node.js: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for .NET: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for C++: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Ruby: [GitHub](#), [Gitee](#)

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

错误码	描述
FailedOperation	操作失败。
InvalidParameter	参数错误。
InvalidParameter.PaginationTokenInvalid	PaginationToken参数非法。
InvalidParameterValue.ReservedTagKey	系统预留标签键 qcloud、tencent和project 禁止创建。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数错误。

获取标签列表

最近更新时间：2025-04-25 01:54:28

1. 接口描述

接口请求域名：tag.tencentcloudapi.com。

用于获取已建立的标签列表。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

</> 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
Action	是	String	公共参数 ，本接口取值：GetTags。
Version	是	String	公共参数 ，本接口取值：2018-08-13。
Region	否	String	公共参数 ，本接口不需要传递此参数。
PaginationToken	否	String	从上一页的响应中获取的下一页的Token值。 如果是第一次请求，设置为空。 示例值：POzXXYGXh2fRDHxx7PGaJnLApcyluAYBC5HX25gKI4Sz2R*
MaxResults	否	Integer	每一页返回的数据最大条数，最大1000。 缺省值：50。 示例值：50
TagKeys.N	否	Array of String	标签键。 返回所有标签键列表对应的标签。 最大长度：20 示例值：["key1"]
Category	否	String	标签类型。取值：Custom：自定义标签。System：系统标签。All：全部标签。默认值：All。 示例值：All

3. 输出参数

参数名称	类型	描述
PaginationToken	String	获取的下一页的Token值，如果当前是最后一页，返回为空 示例值：POzXXYGXh2fRDHxx7PGaJnLApcyluAYBC5HX25gKI4Sz2R*
Tags	Array of Tag	标签列表。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询标签列表

输入示例

```
POST / HTTP/1.1
Host: tag.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: GetTags
<公共请求参数>

{
  "PaginationToken": "Ab9jDQQcWdAXTyU2IUabQzkIK1tLrX19nJCB8MTuJD9Ck8*****",
  "MaxResults": "2"
}
```

输出示例

```
{
  "Response": {
    "PaginationToken": "t1eWpVGSKbAxvaIwRPttZIQq-Eu7e2hWKvN5iMyULmtF*****",
    "Tags": [
      {
        "TagKey": "key1",
        "TagValue": "value1"
      },
      {
        "TagKey": "key2",
        "TagValue": "value2"
      }
    ],
    "RequestId": "0ba573a8-0fdc-44eb-96d7-8f7f*****"
  }
}
```

5. 开发者资源

腾讯云 API 平台

[腾讯云 API 平台](#) 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 [API Inspector](#) 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 [API Explorer](#) 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: [GitHub](#), [Gitee](#)

- Tencent Cloud SDK 3.0 for Java: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for PHP: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Go: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Node.js: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for .NET: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for C++: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Ruby: [GitHub](#), [Gitee](#)

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

错误码	描述
FailedOperation	操作失败。
InvalidParameter	参数错误。
InvalidParameter.PaginationTokenInvalid	PaginationToken参数非法。
InvalidParameter.ReservedTagKey	系统预留标签键 qcloud、tencent和project 禁止创建。
InvalidParameterValue.TagKeyEmpty	标签键不能为空值。
LimitExceeded	超过配额限制。

项目相关接口

获取项目列表

最近更新时间：2025-04-25 01:54:27

1. 接口描述

接口请求域名：tag.tencentcloudapi.com。

获取项目列表

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

<> 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
Action	是	String	公共参数 ，本接口取值：DescribeProjects。
Version	是	String	公共参数 ，本接口取值：2018-08-13。
Region	否	String	公共参数 ，本接口不需要传递此参数。
AllList	是	Integer	传1拉取所有项目（包括隐藏项目），传0拉取显示项目 示例值：1
Limit	是	Integer	分页条数，固定值1000。 示例值：1000
Offset	是	Integer	分页偏移量。 示例值：0
ProjectId	否	Integer	按项目ID筛选，大于0 示例值：1001
ProjectName	否	String	按项目名称筛选 示例值：测试项目

3. 输出参数

参数名称	类型	描述
Total	Integer	数据总条数 示例值：1000
Projects	Array of Project	项目列表
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 获取项目列表

输入示例

```
POST / HTTP/1.1
Host: tag.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DescribeProjects
<公共请求参数>

{
  "Limit": 1000,
  "AllList": 0,
  "Offset": 0
}
```

输出示例

```
{
  "Response": {
    "Total": 1,
    "Projects": [
      {
        "ProjectId": 1000101,
        "ProjectName": "test1",
        "CreatorUin": 100000012345,
        "ProjectInfo": "",
        "CreateTime": "2021-10-13 11:03:30"
      }
    ],
    "RequestId": "d75bbc07-f5bd-4020-b4bd-f877e9ccd1db"
  }
}
```

5. 开发者资源

腾讯云 API 平台

[腾讯云 API 平台](#) 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 [API Inspector](#) 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 [API Explorer](#) 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: [GitHub](#), [Gitee](#)

- Tencent Cloud SDK 3.0 for Java: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for PHP: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Go: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Node.js: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for .NET: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for C++: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Ruby: [GitHub](#), [Gitee](#)

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
AuthFailure.UnauthorizedOperation	未通过CAM鉴权。
FailedOperation	操作失败。
InvalidParameter	参数错误。

创建项目

最近更新时间：2025-04-25 01:54:28

1. 接口描述

接口请求域名：tag.tencentcloudapi.com。

创建项目

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

</> 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
Action	是	String	公共参数 ，本接口取值：AddProject。
Version	是	String	公共参数 ，本接口取值：2018-08-13。
Region	否	String	公共参数 ，本接口不需要传递此参数。
ProjectName	是	String	项目名称 示例值：测试项目
Info	否	String	项目描述 示例值：项目描述

3. 输出参数

参数名称	类型	描述
ProjectId	Integer	项目Id 示例值：1001
IsNew	Integer	是否为新项目，1是新项目，0不是新项目 示例值：1
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 创建项目

创建项目

输入示例

```
POST / HTTP/1.1
Host: tag.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: AddProject
<公共请求参数>

{
  "ProjectName": "测试项目"
}
```

输出示例

```
{
  "Response": {
    "IsNew": 1,
    "ProjectId": 1001,
    "RequestId": "7f831036-30b5-4972-a729-e6851a15d345"
  }
}
```

5. 开发者资源

腾讯云 API 平台

[腾讯云 API 平台](#) 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 [API Inspector](#) 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 [API Explorer](#) 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Java: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for PHP: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Go: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Node.js: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for .NET: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for C++: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Ruby: [GitHub](#), [Gitee](#)

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

错误码	描述
FailedOperation	操作失败。
FailedOperation.ProjectNumExceed	项目数超过限制。
InvalidParameter.ProjectNameExisted	项目名称已存在。
ResourceNotFound.RecordNotFound	记录不存在。

修改项目

最近更新时间：2025-04-25 01:54:27

1. 接口描述

接口请求域名：tag.tencentcloudapi.com。

修改项目

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

</> 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
Action	是	String	公共参数 ，本接口取值：UpdateProject。
Version	是	String	公共参数 ，本接口取值：2018-08-13。
Region	否	String	公共参数 ，本接口不需要传递此参数。
ProjectId	是	Integer	项目ID 示例值：1001
ProjectName	否	String	项目名称 示例值：测试项目
Disable	否	Integer	禁用项目，1，禁用，0，启用 示例值：0
Info	否	String	备注 示例值：备注信息

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 修改项目名称

修改项目名称

输入示例

```
POST / HTTP/1.1
Host: tag.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: UpdateProject
<公共请求参数>
```

```
{
  "ProjectId": 1001,
  "ProjectName": "项目名称"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "87ccfc15-4c5a-4f0a-b5f1-eae50aad9e9a"
  }
}
```

5. 开发者资源

腾讯云 API 平台

[腾讯云 API 平台](#) 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 [API Inspector](#) 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 [API Explorer](#) 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Java: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for PHP: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Go: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Node.js: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for .NET: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for C++: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Ruby: [GitHub](#), [Gitee](#)

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

错误码	描述
FailedOperation.ProjectDisabled	项目已禁用。
FailedOperation.ProjectNumExceed	项目数超过限制。
InvalidParameter	参数错误。
InvalidParameter.ProjectNameExisted	项目名称已存在。
ResourceNotFound.RecordNotFound	记录不存在。

数据结构

最近更新时间：2025-05-28 01:26:39

FailedResource

失败资源信息。
绑定或解绑资源标签时失败返回

被如下接口引用：TagResources, UnTagResources。

名称	类型	描述
Resource	String	失败的资源六段式 示例值：qcs::cvm:ap-beijing:uin/1234567:instance/ins-123
Code	String	错误码 示例值：InvalidParameterValue.TagKeyDuplicate
Message	String	错误信息 示例值：TagKeyDuplicate

Project

项目信息
被如下接口引用：DescribeProjects。

名称	类型	描述
ProjectId	Integer	项目ID 示例值：1
ProjectName	String	项目名称 示例值：project_1
CreatorUin	Integer	创建人uin 示例值：1
ProjectInfo	String	项目描述 示例值：info
CreateTime	String	创建时间 示例值：2022-08-03 00:00:00

ResourceIdTag

资源标签键值
被如下接口引用：DescribeResourceTagsByTagKeys。

名称	类型	描述
ResourceId	String	资源唯一标识 示例值：ins-3841
TagKeyValues	Array of Tag	标签键值对

ResourceTag

资源标签

被如下接口引用：DescribeResourcesByTags, DescribeResourcesByTagsUnion。

名称	类型	描述
ResourceRegion	String	资源所在地域 示例值：ap-guangzhou
ServiceType	String	业务类型 示例值：cvm
ResourcePrefix	String	资源前缀 示例值：instance
ResourceId	String	资源唯一标记 示例值：ins-1399
Tags	Array of Tag	资源标签

ResourceTagMapping

资源及关联的标签(键和值)。

被如下接口引用：GetResources。

名称	类型	描述
Resource	String	资源六段式。腾讯云使用资源六段式描述一个资源。 例如：ResourceList.1 = qcs::\${ServiceType}:\${Region}:\${Account}:\${ResourcePreifx}/\${ResourceId}。 示例值：qcs::cvm:ap-beijing:uin/1234567:instance/ins-123
Tags	Array of Tag	资源关联的标签列表

Tag

表示一个标签键值对

被如下接口引用：CreateTags, DeleteTags, DescribeResourceTagsByTagKeys, DescribeResourcesByTags, DescribeResourcesByTagsUnion, DescribeTagValues, DescribeTagValuesSeq, GetResources, GetTagValues, GetTags, ModifyResourceTags, TagResources。

名称	类型	必选	描述
TagKey	String	是	标签键 示例值：name
TagValue	String	是	标签值 示例值：John
Category	String	否	标签类型。取值： Custom：自定义标签。 System：系统标签。 All：全部标签。默认值：All。 示例值：All

TagFilter

tag过滤数组多个是与的关系

被如下接口引用：DescribeResourcesByTags, DescribeResourcesByTagsUnion, GetResources。

名称	类型	必选	描述
TagKey	String	是	标签键 示例值：name
TagValue	Array of String	否	标签值数组 多个值的话是或的关系 示例值：["john"]

TagKeyObject

标签键对象

被如下接口引用：ModifyResourceTags。

名称	类型	必选	描述
TagKey	String	是	标签键 示例值：name

TagResource

标签键值对以及资源ID

被如下接口引用：DescribeResourceTags, DescribeResourceTagsByResourceIds, DescribeResourceTagsByResourceIdsSeq。

名称	类型	描述
TagKey	String	标签键 示例值：name
TagValue	String	标签值 示例值：john
ResourceId	String	资源ID 示例值：ins-2568
TagKeyMd5	String	标签键MD5值 示例值：b068931cc450442b63f5b3d276ea4297
TagValueMd5	String	标签值MD5值 示例值：527bd5b5d689e2c32ae974c6229ff785
ServiceType	String	资源类型 示例值：cvm
Category	String	标签类型。取值：Custom：自定义标签。System：系统标签。 示例值：Custom

TagWithDelete

表示一个标签键值对以及是否允许删除

被如下接口引用：DescribeTags, DescribeTagsSeq。

名称	类型	描述
TagKey	String	标签键 示例值：name

名称	类型	描述
TagValue	String	标签值 示例值：john
CanDelete	Integer	是否可以删除 示例值：0
Category	String	标签类型。取值：Custom：自定义标签。System：系统标签。All：全部标签。默认值：All。 示例值：Custom

错误码

最近更新时间：2024-12-20 01:52:53

功能说明

如果返回结果中存在 Error 字段，则表示调用 API 接口失败。例如：

```
{
  "Response": {
    "Error": {
      "Code": "AuthFailure.SignatureFailure",
      "Message": "The provided credentials could not be validated. Please check your signature is correct."
    },
    "RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
  }
}
```

Error 中的 Code 表示错误码，Message 表示该错误的具体信息。

错误码列表

公共错误码

错误码	说明
ActionOffline	接口已下线。
AuthFailure.InvalidAuthorization	请求头部的 <code>Authorization</code> 不符合腾讯云标准。
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）。
AuthFailure.MFAFailure	MFA 错误。
AuthFailure.SecretIdNotFound	密钥不存在。请在 控制台 检查密钥是否已被删除或者禁用，如状态正常，请检查密钥是否填写正确，注意前后不得有空格。
AuthFailure.SignatureExpire	签名过期。Timestamp 和服务器时间相差不得超过五分钟，请检查本地时间是否和标准时间同步。
AuthFailure.SignatureFailure	签名错误。签名计算错误，请对照调用方式中的签名方法文档检查签名计算过程。
AuthFailure.TokenFailure	token 错误。
AuthFailure.UnauthorizedOperation	请求未授权。请参考 CAM 文档对鉴权的说明。
DryRunOperation	DryRun 操作，代表请求将会是成功的，只是多传了 DryRun 参数。
FailedOperation	操作失败。
InternalError	内部错误。
InvalidAction	接口不存在。
InvalidParameter	参数错误（包括参数格式、类型等错误）。
InvalidParameterValue	参数取值错误。

错误码	说明
InvalidRequest	请求 body 的 multipart 格式错误。
IpInBlacklist	IP 地址在黑名单中。
IpNotInWhitelist	IP 地址不在白名单中。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数。
NoSuchProduct	产品不存在
NoSuchVersion	接口版本不存在。
RequestLimitExceeded	请求的次数超过了频率限制。
RequestLimitExceeded.GlobalRegionUinLimitExceeded	主账号超过频率限制。
RequestLimitExceeded.IPLimitExceeded	IP 限频。
RequestLimitExceeded.UinLimitExceeded	主账号限频。
RequestSizeLimitExceeded	请求包超过限制大小。
ResourceInUse	资源被占用。
ResourceInsufficient	资源不足。
ResourceNotFound	资源不存在。
ResourceUnavailable	资源不可用。
ResponseSizeLimitExceeded	返回包超过限制大小。
ServiceUnavailable	当前服务暂时不可用。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误，用户多传未定义的参数会导致错误。
UnsupportedOperation	操作不支持。
UnsupportedProtocol	http(s) 请求协议错误，只支持 GET 和 POST 请求。
UnsupportedRegion	接口不支持所传地域。

业务错误码

错误码	说明
AuthFailure	CAM签名/鉴权错误。
FailedOperation.PolicyEnforcedInvalid	标签策略强制打标签限制
FailedOperation.ProjectDisabled	项目已禁用。
FailedOperation.ProjectNumExceed	项目数超过限制。
FailedOperation.ResourceAppldNotSame	单次请求的资源appld必须相同。
FailedOperation.ResourceTagProcessing	资源标签正在处理中。

错误码	说明
FailedOperation.TagAttachedQuota	标签已经关联配额。
FailedOperation.TagAttachedResource	已关联资源的标签无法删除。
InvalidParameter.PaginationTokenInvalid	PaginationToken参数非法。
InvalidParameter.ProjectNameExisted	项目名称已存在。
InvalidParameter.ReservedTagKey	系统预留标签键 qcloud、tencent和project 禁止创建。
InvalidParameter.Tag	Tag参数错误。
InvalidParameter.UnsupportedService	当前业务不支持标签操作。
InvalidParameterValue.DeleteTagsParamError	DeleteTags中不能包含ReplaceTags或AddTags中元素。
InvalidParameterValue.OffsetInvalid	offset error。
InvalidParameterValue.RegionInvalid	地域错误。
InvalidParameterValue.ReservedTagKey	系统预留标签键 qcloud、tencent和project 禁止创建。
InvalidParameterValue.ResourceDescriptionError	资源描述错误。
InvalidParameterValue.ResourceIdInvalid	资源Id错误。
InvalidParameterValue.ResourcePrefixInvalid	资源前缀错误。
InvalidParameterValue.ServiceTypeInvalid	业务类型错误。
InvalidParameterValue.TagFilters	TagFilters参数错误。
InvalidParameterValue.TagFiltersLengthExceeded	过滤标签键对应标签值达到上限数 6。
InvalidParameterValue.TagKeyCharacterIllegal	标签键包含非法字符。
InvalidParameterValue.TagKeyDuplicate	TagList中存在重复的TagKey。
InvalidParameterValue.TagKeyEmpty	标签键不能为空值。
InvalidParameterValue.TagKeyLengthExceeded	标签键长度超过限制。
InvalidParameterValue.TagValueCharacterIllegal	标签值包含非法字符。
InvalidParameterValue.TagValueEmpty	标签值不能为空值。
InvalidParameterValue.TagValueLengthExceeded	标签值长度超过限制。
InvalidParameterValue.UinInvalid	Uin参数不合法。
LimitExceeded.Quota	超过配额限制。
LimitExceeded.ResourceAttachedTags	资源关联的标签数超过限制。
LimitExceeded.ResourceNumPerRequest	单次请求的资源数达到上限。
LimitExceeded.TagKey	用户创建标签键达到上限数 1000。
LimitExceeded.TagNumPerRequest	单次请求的标签数超过上限。
LimitExceeded.TagValue	单个标签键对应标签值达到上限数 1000。
OperationDenied	操作被拒绝。

错误码	说明
ResourceInUse.TagDuplicate	标签已存在。
ResourceInUse.TagKeyAttached	对应的标签键和资源已关联。
ResourceNotFound.AttachedTagKeyNotFound	资源关联的标签键不存在。
ResourceNotFound.RecordNotFound	记录不存在。
ResourceNotFound.TagNonExist	标签不存在。

标签 API 2017

API 概览

最近更新时间：2019-09-12 14:47:09

写接口

接口名称	接口功能
CreateTag	创建标签
DeleteTag	删除标签
UpdateResourceTagValue	修改标签资源值
ModifyResourceTags	批量操作资源所有标签
AddResourceTag	标签关联资源
DeleteResourceTag	资源标签解绑

读接口

接口名称	接口功能
GetTagValues	查询标签值
GetTags	查询标签列表
GetTagKeys	查询标签键
GetResourceTagsByResourceIds	查看资源关联的标签

调用方式

请求结构

最近更新时间：2019-09-12 14:48:02

对腾讯云的 API 接口的调用是通过向腾讯云 API 的服务端地址发送请求，并按照接口说明在请求中加入相应的请求参数来完成的。腾讯云 API 的请求结构由：服务地址、通信协议、请求方法、请求参数和字符编码组成。具体描述如下：

服务地址

腾讯云 API 的服务接入地址与具体模块相关，详细请参见各接口相关描述。

通信协议

腾讯云 API 的大部分接口都通过 HTTPS 进行通信，为您提供高安全性的通信通道。

请求方法

腾讯云 API 同时支持 POST 和 GET 请求。

⚠ 注意：

1. POST 和 GET 请求不能混合使用，若使用 GET 方式，则参数均从 Querystring 取得；若使用 POST 方式，则参数均从 Request Body 中取得，而 Querystring 中的参数将忽略。两种请求方式的参数格式规则相同，一般情况下使用 GET 请求，当参数字符串过长时推荐使用 POST。
2. 如果用户的请求方法是 GET，则对所有请求参数值均需要做 URL 编码，若为 POST，则无需对参数编码。
3. GET 请求的最大长度根据不同的浏览器和服务器设置有所不同，例如，传统 IE 浏览器限制为2K，Firefox 限制为 8K，对于一些参数较多、长度较长的 API 请求，建议您使用 POST 方法以免在请求过程中会由于字符串超过最大长度而导致请求失败。
4. 对于 POST 请求，您需要使用 `x-www-form-urlencoded` 的形式传参，因为云 API 侧是从 `$_POST` 中取出请求参数的。

请求参数

腾讯云 API 的每个请求都需要指定两类参数：公共请求参数以及接口请求参数。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 [公共请求参数](#)，而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。

字符编码

腾讯云 API 的请求及返回结果均使用 UTF-8 字符集进行编码。

返回结果

最近更新时间：2020-02-13 14:27:46

❗ 说明：

当前页面接口为旧版 API，未来可能停止维护，目前不展示在左侧导航。云服务器 API 3.0 版本接口定义更加规范，访问时延下降显著，建议使用 [云服务器 API 3.0](#)。

若 API 调用成功，则最终返回结果中的错误码 code 为 0，错误信息 message 为空，并且会显示返回的结果数据。

示例如下：

```
{
  "code": "0",
  "message": ""
  <返回结果数据>
}
```

公共参数

最近更新时间：2020-02-11 17:24:28

一个完整的腾讯云 API 请求需要两类请求参数：公共请求参数和接口请求参数。本文将介绍腾讯云 API 请求需要用到的6个公共请求参数，有关接口请求参数的详细说明请参见 [接口请求参数](#) 章节。

公共请求参数是每个接口都需要使用到的请求参数，开发者每次使用腾讯云 API 发送请求时都需要携带这些公共请求参数，否则会导致请求失败。公共请求参数的首字母均为大写，以此区分于接口请求参数。

公共请求参数具体列表如下：

⚠ 注意：

文章中的接口实例以腾讯云 CVM 为例，各腾讯云产品的具体使用方式请对应实际产品。

参数名称	描述	类型	必选
Action	具体操作的指令接口名称，例如腾讯云 CVM 用户调用 查询实例列表 接口，则 Action 参数即为 DescribeInstances。	String	是
Region	地域参数，用来标识希望操作哪个地域的实例。详细信息可参见 地域和可用区 列表，或使用 查询地域列表 API 接口查看。 注意： 1. 正常情况下此参数是必须的，如无需传入，则会在相应接口中进行说明； 2. 部分区域正在内测中，目前仅面向部分用户开放。	String	否
Timestamp	当前 UNIX 时间戳，可记录发起 API 请求的时间。	UInt	是
Nonce	用户可自定义随机正整数，与 Timestamp 联合起来，用于防止重放攻击。	UInt	是
SecretId	在 云API密钥 上申请的标识身份的 SecretId，一个 SecretID 对应唯一的 SecretKey，而 SecretKey 会用来生成请求签名 Signature。具体可参考 签名方法 章节。	String	是
Signature	请求签名，用来验证此次请求的合法性，需要用户根据实际的输入参数计算得出。计算方法可参考 签名方法 章节。	String	是
SignatureMethod	签名方式，目前支持 HmacSHA256 和 HmacSHA1。只有指定此参数为 HmacSHA256 时，才使用 HmacSHA256 算法验证签名，其他情况均使用 HmacSHA1 验证签名。详细签名计算方法可参考 签名方法 章节。	String	否
Token	临时证书所用的 Token，需要结合临时密钥一起使用。长期密钥不需要 Token。	String	否

使用示例

腾讯各云产品 API 请求链接中，公共请求参数的形式如下，以腾讯云 CVM 为例，假设用户想要查询广州地域的云服务器实例列表，则其请求链接的形式为：

```
https://cvm.api.qcloud.com/v2/index.php?
Action=DescribeInstances
&SecretId=xxxxxxx
&Region=ap-guangzhou
&Timestamp=1465055529
&Nonce=59485
&Signature=mysignature
&SignatureMethod=HmacSHA256
&<接口请求参数>
```

请求签名

最近更新时间：2024-11-18 14:32:33

⚠ 注意：

- 腾讯云 NLP 服务已于2019年07月09日全新升级为 **新版**，接口功能更全面，服务更加稳定，且公测期间免费使用。
- 老版本接口将不再继续维护，将于2019年11月16日零点下线，建议您使用 **新版 API**，体验更优服务。
- 接口切换过程中，若您有相关问题，可加入官方 QQ 群（330130409）详细咨询。

腾讯云 API 会对每个访问的请求进行身份验证，即每个请求都需要在公共请求参数中包含签名信息（Signature），以验证用户身份。签名信息由用户所执有的安全凭证生成，安全凭证包括 SecretId 和 SecretKey，若用户还没有安全凭证，则需在腾讯云官网上自主申请，否则无法调用云 API 接口。

申请安全凭证

在第一次使用腾讯云 API 之前，用户需在[腾讯云控制台](#) > [API 密钥管理](#) 上申请安全凭证。安全凭证包括 SecretId 和 SecretKey，其中：

- SecretId**：用于标识 API 调用者身份。
- SecretKey**：用于加密签名字符串和服务器端验证签名字符串的密钥。

⚠ 注意：

API 密钥是构建腾讯云 API 请求的重要凭证，使用腾讯云 API 可以操作您名下的所有腾讯云资源，为了您的财产和服务安全，请妥善保管并定期更换密钥，当您更换密钥后，请及时删除旧密钥。

申请安全凭证步骤：

- 登录腾讯云控制台，进入 [API 密钥管理](#) 页面。
- 在 API 密钥管理页面，单击**新建密钥**即可以创建一对 SecretId/SecretKey。

⚠ 注意：

- 开发商账号最多可以拥有两对 SecretId / SecretKey。
- 被开发商添加为子用户的 QQ 账号，在不同开发商控制台，可以申请不同的安全凭证。
- 子用户的安全凭证，目前仅可调用部分接口的云 API。

生成签名串

有了安全凭证 SecretId 和 SecretKey 后，就可以生成签名串了。生成签名串的详细过程如下：



假设用户的 SecretId 和 SecretKey 分别是：

SecretId: *****

SecretKey: *****

⚠ 注意：

这里只是示例，请用户根据自己实际的 SecretId 和 SecretKey 和请求参数进行后续操作。

以腾讯云 CVM 为例，当用户调用腾讯云 CVM 的 [查看实例列表](#) (DescribeInstances)接口时，其请求参数为：

参数名称	描述	参数值
Action	方法名	DescribeInstances

SecretId	密钥 ID	*****
Timestamp	当前时间戳	1465185768
Nonce	随机正整数	11886
Region	实例所在区域	ap-guangzhou
SignatureMethod	签名方式	HmacSHA256
InstanceId.0	待查询的实例 ID	ins-09dx96dg

1. 对参数排序

首先对所有请求参数按参数名做字典序升序排列。（所谓字典序升序排列，直观上就如同在字典中排列单词一样排序，按照字母表或数字表里递增顺序的排列次序，即先考虑第一个“字母”，在相同的情况下考虑第二个“字母”，依此类推。）您可以借助编程语言中的相关排序函数来实现这一功能，如 PHP 中的 ksort 函数。上述示例参数的排序结果如下：

```
{
  "Action" : "DescribeInstances",
  "InstanceIds.0" : "ins-09dx96dg",
  "Nonce" : "11886",
  "Region" : "ap-guangzhou",
  "SecretId" : "*****",
  "SignatureMethod" : "HmacSHA256",
  "Timestamp" : "1465185768"
}
```

使用其它程序设计语言开发时，可对上面示例中的参数进行排序，得到的结果一致即可。

2. 拼接请求字符串

此步骤将生成请求字符串。

将把上一步排序好的请求参数格式化成“参数名称”=“参数值”的形式，如对 Action 参数，其参数名称为 "Action"，参数值为 "DescribeInstances"，因此格式化后就为 Action=DescribeInstances。

⚠ 注意：

- “参数值”为原始值而非 URL 编码后的值。
- 若输入参数的 Key 中包含下划线，则需要将其转换为 .，但是 Value 中的下划线则不用转换。如 Placement_Zone=CN_GUANGZHOU，则需要将其转换成 Placement.Zone=CN_GUANGZHOU。

然后将格式化后的各个参数用 "&" 拼接在一起，最终生成的请求字符串为（请忽略文中的换行）：

```
Action=DescribeInstances
&InstanceIds.0=ins-09dx96dg
&Nonce=11886
&Region=ap-guangzhou
&SecretId=*****
&SignatureMethod=HmacSHA256
&Timestamp=1465185768
```

3. 拼接签名原文字符串

此步骤将生成签名原文字符串。

签名原文字符串的拼接规则为：

请求方法 + 请求主机 + 请求路径 + ? + 请求字符串

参数构成说明：

- **请求方法**：支持 POST 和 GET 方式，这里使用 GET 请求，注意方法为全大写。
- **请求主机**：即主机域名，请求域名由接口所属的产品或所属产品的模块决定，不同产品或不同产品的模块的请求域名会有不同。如腾讯云 CVM 的查询实例列表（DescribeInstances）的请求域名为：`cvm.api.qcloud.com`，具体产品请求域名详见各接口说明。
- **请求路径**：腾讯云 API 对应产品的请求路径，一般是一个产品对应一个固定路径，如腾讯云 CVM 请求路径固定为 `/v2/index.php`。
- **请求字符串**：即上一步生成的请求字符串。

因此，上述示例的拼接签名原文字符串结果为（请忽略文中的换行）：

```
GETcvm.api.qcloud.com/v2/index.php?Action=DescribeInstances
&InstanceIds.0=ins-09dx96dg
&Nonce=11886
&Region=ap-guangzhou
&SecretId=*****
&SignatureMethod=HmacSHA256
&Timestamp=1465185768
```

4. 生成签名串

此步骤生成签名串。

⚠ 注意：

计算签名的方法有两种：HmacSHA256 和 HmacSHA1 这里要根据您指定的签名算法（即 SignatureMethod 参数）生成签名串。当指定 SignatureMethod 为 HmacSHA256 时，需要使用 HmacSHA256 计算签名，其他情况请使用 HmacSHA1 计算签名。

首先使用签名算法（HmacSHA256 或 HmacSHA1）对上一步中获得的 **签名原文字符串** 进行签名，然后将生成的签名串使用 Base64 进行编码，即可获得最终的签名串。

具体代码如下，以 PHP 语言为例，由于本例中所用的签名算法为 **HmacSHA256**，因此生成签名串的代码如下（使用其它程序设计语言开发时，可用上述示例中的原文字符串进行签名验证，得到的签名串与例子中的一致即可）：

```
$secretKey = '*****';
$srcStr = 'GETcvm.api.qcloud.com/v2/index.php?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Nonce=11886&Region=ap-guangzhou&SecretId=*****&SignatureMethod=HmacSHA256&Timestamp=1465185768';
$signStr = base64_encode(hash_hmac('sha256', $srcStr, $secretKey, true));
echo $signStr;
```

最终得到的签名串为：

```
0EEh/HtGRr/VJXTAD9tYMth1Bzm3lLHz5RCDv1GdM8s=
```

同理，当您指定签名算法为 **HmacSHA1** 时，生成签名串的代码如下：

```
$secretKey = '*****';
$srcStr = 'GETcvm.api.qcloud.com/v2/index.php?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Nonce=11886&Region=ap-guangzhou&SecretId=*****&SignatureMethod=HmacSHA1&Timestamp=1465185768';
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));
echo $signStr;
```

最终得到的签名串为：

nPVnY6njQmwQ8ciqbPl5Qe+Oru4=

签名串编码

生成的签名串并不能直接作为请求参数，需要对其进行 URL 编码。

如上一步生成的签名串为 0EEem/HtGRr/VJXTAD9tYMth1Bzm3lLHz5RCDv1GdM8s= ，则其编码后为 0EEem%2FHtGRr%2FVJXTAD9tYMth1Bzm3lLHz5RCDv1GdM8s%3D 。因此，最终得到的签名串请求参数 (Signature) 为： 0EEem%2FHtGRr%2FVJXTAD9tYMth1Bzm3lLHz5RCDv1GdM8s%3D ，它将用于生成最终的请求URL。

 **注意：**

如果用户的请求方法是 GET，则对所有请求参数的参数值均需要做 URL 编码；此外，部分语言库会自动对 URL 进行编码，重复编码会导致签名校验失败。

鉴权失败

当鉴权不通过时，可能出现如下表的错误：

错误代码	错误类型	错误描述
4100	身份认证失败	身份验证失败，请确保您请求参数中的 Signature 按照上述步骤计算正确，特别注意 Signature 要做 url 编码后再发起请求。
4101	未被开发商授权访问本接口	该子用户未被授权调用此接口。请联系开发商授权，详情请查阅 授权策略 。
4102	未被开发商授权访问本接口中所操作的资源	请求的资源参数中，存在未被开发商授权访问的资源，请在 message 字段中查看无权查看的资源 ID。 请联系开发商授权，详情请查阅 授权策略 。
4103	非开发商的 SecretId 暂不支持调用本接口	子用户的 SecretId 不支持调用此接口，只有开发商有权调用。
4104	SecretId 不存在	签名所用的 SecretId 不存在，也可能是密钥状态有误，请确保 API 密钥有效且未被禁用。
4110	鉴权失败	权限校验失败，请确保您有使用所访问资源的权限。
4500	重放攻击错误	请注意 Nonce 参数两次请求不能重复，Timestamp 与腾讯服务器相差不能超过2小时。

写接口

创建标签

最近更新时间：2019-09-12 14:52:12

接口描述

创建标签 CreateTag 用于对腾讯云上资源进行标签创建操作。

请求域名：tag.api.qcloud.com

输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
tagKey	是	String	标签键
tagValue	是	String	标签值

输出参数

无

错误码

错误码	描述
30001	标签键不存在
30002	系统预留标签键 qcloud: 和 tencent: 禁止创建
30004	主账号 Uin 不存在
30016	创建者 Uin 不存在
30017	用户标签已存在
30010	用户创建标签键达到上限数 1000
30020	单个标签键对应标签值达到上限数 1000

示例

输入示例

```
https://domain/v2/index.php?Action=CreateTag&tagKey=t1&tagValue=v1<公共请求参数>
```

输出示例

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success",
  "data": []
}
```

修改资源标签值

最近更新时间：2019-09-12 14:51:07

接口描述

修改资源标签值 UpdateResourceTagValue 用于修改腾讯云上资源已关联标签的标签值（标签键不变）。

请求域名：`tag.api.qcloud.com`

输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
tagKey	是	String	资源关联的标签键
tagValue	是	String	修改后的标签值
resource	是	String	资源六段式标记

输出参数

无

错误码

错误码	描述
30001	标签键不存在
30006	标签值不存在
30004	主账号 Uin 不存在
30016	创建者 Uin 不存在
30003	资源六段式标记参数错误
30007	资源标签不存在
30020	单个标签键对应标签值达到上限数 1000

示例

输入示例

```
https://domain/v2/index.php?
Action=UpdateResourceTagValue&tagKey=t1&tagValue=v1&resource=qcs::cvm:ap-
beijing:uin/1234567:instance/ins-123&<公共请求参数>
```

输出示例

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success",
  "data": []
}
```

批量操作资源所有标签

最近更新时间：2019-06-26 15:31:58

接口描述

批量操作资源所有标签 ModifyResourceTags 用于对腾讯云上已有资源进行批量标签操作。

请求域名：`tag.api.qcloud.com`

输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
resource	是	String	资源六段式标记
replaceTags	否	Array	需要修改的标签集合
deleteTags	否	Array	需要删除的标签集合

其中 replaceTags 数组的参数如下：

参数名称	必选	类型	描述
tagKey	否	String	标签键，replaceTags 中的元素
tagValue	否	String	标签值，replaceTags 中的元素

其中 deleteTags 数组的参数如下：

参数名称	必选	类型	描述
tagKey	否	String	标签键，deleteTags 中的元素
tagValue	否	String	标签值，deleteTags 中的元素

输出参数

无。

错误码

错误码	描述
9003	参数错误
30001	标签键不存在
30006	标签值不存在
30004	主账号 Uin 不存在
30016	创建者 Uin 不存在
30003	资源六段式标记参数错误
30007	资源标签不存在
30010	单个用户最多1000个不同的 key
30011	单个资源标签键数不能超过50

30020

单个标签键对应标签值达到上限数1000

示例

输入示例

```
https://domain/v2/index.php?
Action=ModifyResourceTags&replaceTags.0.tagKey=t1&replaceTags.0.tagValue=v1&deleteTags.0.tagKey=t2&
replaceTags.0.tagValue=v2&resource=qcs::cvm:ap-beijing:uin/1234567:instance/ins-123&<公共请求参数>
```

输出示例

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success",
  "data": []
}
```

标签关联资源

最近更新时间：2019-11-19 15:37:19

接口描述

标签关联资源 AddResourceTag 用于对腾讯云上资源进行关联操作。

请求域名：`tag.api.qcloud.com`

输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
tagKey	是	String	标签键
tagValue	是	String	标签值
resource	是	String	资源六段式标记

输出参数

无

错误码

错误码	描述
30001	标签键不存在
30002	系统预留标签键 <code>qcloud:</code> 和 <code>tencent:</code> 禁止创建
30004	主账号 Uin 不存在
30016	创建者 Uin 不存在
30003	资源六段式标记参数错误
30010	用户创建标签键达到上限数1000
30020	单个标签键对应标签值达到上限数1000
30005	用户资源标签已存在不能重复添加

示例

输入示例

```
https://domain/v2/index.php?Action=AddResourceTag&tagKey=t1&tagValue=v1&resource=qcs::cvm:ap-beijing:uin/1234567:instance/ins-123&<公共请求参数>
```

输出示例

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success",
  "data": []
}
```

}

资源标签解绑

最近更新时间：2019-09-12 14:50:31

接口描述

资源标签解绑 DeleteResourceTag 用于对腾讯云上已关联标签的资源进行解绑操作。

请求域名：`tag.api.qcloud.com`

输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
tagKey	是	String	标签键
tagValue	是	String	标签值
resource	是	String	资源六段式标记

输出参数

无

错误码

错误码	描述
30001	标签键不存在
30004	主账号 Uin 不存在
30016	创建者 Uin 不存在
30003	资源六段式标记参数错误
30007	资源标签不存在

示例

输入示例

```
https://domain/v2/index.php?Action=DeleteResourceTag&tagKey=t1&tagValue=v1&resource=qcs::cvm:ap-beijing:uin/1234567:instance/ins-123&<公共请求参数>
```

输出示例

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success",
  "data": []
}
```

删除标签

最近更新时间：2019-09-12 14:49:59

接口描述

删除标签 DeleteTag 用于对腾讯云上资源进行标签删除操作。

请求域名：`tag.api.qcloud.com`

输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
tagKey	是	String	标签键
tagValue	是	String	标签值

输出参数

无

错误码

错误码	描述
30001	标签键不存在
30004	主账号 Uin 不存在
30016	创建者 Uin 不存在

示例

输入示例

```
https://domain/v2/index.php?Action=DeleteTag&tagKey=t1&tagValue=v1&<公共请求参数>
```

输出示例

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success",
  "data": []
}
```

读接口

查询标签列表

最近更新时间：2019-09-12 14:49:28

接口描述

查询标签列表 GetTags 用于查询已建立的标签列表。

请求域名：`tag.api.qcloud.com`

输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
createUin	否	int	创建者用户uin，不传或为空只将uin作为条件查询
page	否	int	页码，默认为 1
rp	否	int	每页大小，默认为 15
tagKey	否	String	标签键,与标签值同时存在或同时不存在，不存在时表示查询该用户所有标签
tagValue	否	String	标签值,与标签键同时存在或同时不存在，不存在时表示查询该用户所有标签

输出参数

参数名称	类型	描述
total	int	结果总数
page	int	当前页码
rp	int	每页大小
rows	Array	标签列表

其中 rows 数组的参数如下：

参数名称	类型	描述
tagKey	String	标签键
tagValue	String	标签值
canDelete	int	能否删除，该标签绑定资源是不能删除

错误码

错误码	描述
30004	主账号 Uin 不存在

示例

输入示例

```
https://domain/v2/index.php?Action=GetTags&tagKey=testTagKey&tagValue=testTagVal&<公共请求参数>
```

输出示例

```
{
  "total": 1,
  "page": "1",
  "rp": "15",
  "rows": [
    {
      "tagKey": "testTagKey",
      "tagValue": "testTagVal",
      "canDelete": 0
    }
  ]
}
```

查询标签键

最近更新时间：2019-09-12 14:48:51

接口描述

查询标签键 GetTagKeys 用于查询已建立的标签列表中的标签键。

请求域名：`tag.api.qcloud.com`

输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
createUin	否	Int	创建者用户 Uin，不传或为空只将 Uin 作为条件查询
page	否	Int	页码，默认为 1
rp	否	Int	每页大小，默认为 15

输出参数

参数名称	类型	描述
total	Int	结果总数
page	Int	当前页码
rp	Int	每页大小
rows	Array	标签列表

其中 rows 数组的参数如下：

参数名称	类型	描述
tagKey	String	标签键

错误码

错误码	描述
30004	主账号 Uin 不存在

示例

输入示例

```
https://domain/v2/index.php?Action=GetTagKeys<公共请求参数>
```

输出示例

```
{
  "total": 1,
  "page": "1",
  "rp": "15",
  "rows": [
```

```
{  
  "tagKey": "testTagKey"  
}
```

查询标签值

最近更新时间：2019-11-19 15:35:54

接口描述

查询标签值 GetTagValues 用于查询已建立的标签列表中的标签值。

请求域名：`tag.api.qcloud.com`

输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
createUin	否	Int	创建者用户 Uin，不传或为空只将 Uin 作为条件查询
tagKeys	是	Array	标签键列表
page	否	Int	页码，默认为1
rp	否	String	每页大小，默认为15

输出参数

参数名称	类型	描述
total	Int	结果总数
page	Int	当前页码
rp	Int	每页大小
rows	Array	标签列表

其中 rows 数组的参数如下：

参数名称	类型	描述
tagKey	String	标签键
tagValue	String	标签值

错误码

错误码	描述
30001	标签键不存在
30004	主账号 Uin 不存在

示例

输入示例

```
https://domain/v2/index.php?Action=GetTags&tagKeys.0=testTagKey1&tagKeys.1=testTagKey2<公共请求参数>
```

输出示例

```
{
  "total": 2,
  "page": "1",
  "rp": "15",
  "rows": [
    {
      "tagKey": "testTagKey1",
      "tagValue": "testTagVal1"
    },
    {
      "tagKey": "testTagKey2",
      "tagValue": "testTagVal2"
    }
  ]
}
```

查看资源关联的标签

最近更新时间：2019-11-19 15:34:21

接口描述

查询资源所有标签键标签值 GetResourceTagsByResourceIds 用于查询已有资源标签键值对。

请求域名：`tag.api.qcloud.com`

输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 [公共请求参数](#)。

参数名称	必选	类型	描述
createUin	否	Int	创建者用户 uin，不传或为空则将当前登录 uin 作为条件查询
page	否	Int	页码，默认为1
rp	否	Int	每页大小，默认为15
region	是	String	资源所在地域
serviceType	是	String	业务类型
resourcePrefix	是	String	资源前缀
resourceIds	是	Array	资源唯一标记

输出参数

data 的参数如下：

参数名称	类型	描述
total	Int	结果总数
page	Int	当前页码
rp	Int	每页大小
rows	Array	标签列表

rows 的参数如下：

参数名称	类型	描述
tagKey	String	标签键
tagValue	String	标签值
resourceId	String	资源唯一标记

错误码

错误码	描述
30004	主账号 Uin 不存在

示例

输入示例

```
https://domain/v2/index.php?Action=GetResourceTagsByResourceIds&region=ap-  
guangzhou&seviceType=cvm&prefix=instance&resourceId.0=ins-123&resourceId.1=ins-456<公共请求参数>
```

输出示例

```
{  
  "total": 4,  
  "page": 1,  
  "rp": 15,  
  "rows": [  
    {  
      "tagKey": "testTagKey1",  
      "tagValue": "testTagVal1",  
      "resourceId": "ins-123"  
    },  
    {  
      "tagKey": "testTagKey2",  
      "tagValue": "testTagVal2",  
      "resourceId": "ins-123"  
    },  
    {  
      "tagKey": "testTagKey3",  
      "tagValue": "testTagVal3",  
      "resourceId": "ins-123"  
    },  
    {  
      "tagKey": "testTagKey1",  
      "tagValue": "testTagVal1",  
      "resourceId": "ins-456"  
    }  
  ]  
}
```

错误码

最近更新时间：2020-02-11 17:22:36

错误码	说明	处理措施
9003, 1000	请求参数有误	请检查参数是否正确，例如 uin 是否是数字，参数要求是数组传递为字符串
30002	权限不允许	对于一些系统保留标签不允许添加
30006	标签值传递有问题	标签值不存在
30010	用户标签 key 已达上限	系统限制，用户标签 key 数量最大支持 1000 个
30011	用户单个资源标签已达上限	系统限制，单个资源标签数最大支持 50 个不同的 key
30012	地域参数错误	地域参数传递错误，请根据现有的地域传递
30013	业务类型错误	业务类型参数传递错误，请根据现有的业务传递
30014	资源前缀错误	资源前缀参数传递错误，请根据现有的业务传递
30020	用户的标签值已达上限	系统限制，用户标签单个 key 最大支持 1000 个 value
30021	标签键有非法字符	可用 UTF-8 格式表示的字母、空格和数字以及特殊字符 <code>+ - = . _ : / @</code> ，标签键长度 - 127 个字符（采用 UTF-8 格式）
30022	标签值有非法字符	可用 UTF-8 格式表示的字母、空格和数字以及特殊字符 <code>+ - = . _ : / @</code> ，标签值长度 - 255 个字符（采用 UTF-8 格式）
30030	标签过滤参数传递错误	未传递标签过滤参数
1000	记录不存在	-
1006	用户不存在	-
1007	用户非法	-
1015	用户创建的项目已达最大限制	超过系统限制 500 个
1036	项目名称已经存在	请更换项目名称
1072	项目不可用	项目已被禁用