

# 全栈式风控引擎 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

控制台相关接口

创建黑白名单

删除黑白名单

查询黑白名单

查询黑白名单数据

客户资源用量查询

添加黑白名单数据

修改黑白名单

修改黑白名单数据

查询黑白名单详情

删除黑白名单数据

API相关接口

全栈式风控引擎

数据结构

错误码

# API文档

## 更新历史

最近更新时间：2025-03-28 14:27:44

### 第 16 次发布

发布时间：2025-03-28 14:27:40

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- [DataAuthorizationInfo](#)
  - 删除成员：IsPersonalData

### 第 15 次发布

发布时间：2024-11-19 01:52:44

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [DescribeUserUsageCnt](#)

新增数据结构：

- [OutputDescribeUserUsageCnt](#)
- [OutputDescribeUserUsageCntData](#)

### 第 14 次发布

发布时间：2024-09-25 01:29:30

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- [DataAuthorizationInfo](#)
  - 新增成员：IsOrderHandling
  - 修改成员：IsPersonalData

### 第 13 次发布

发布时间：2024-08-01 02:05:51

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- [InputImportNameListDataFront](#)
  - 删除成员：FileCode, DataContent

### 第 12 次发布

发布时间：2024-07-31 11:55:08

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [CreateNameList](#)
- [DeleteNameList](#)
- [DeleteNameListData](#)
- [DescribeNameList](#)
- [DescribeNameListDataList](#)
- [DescribeNameListDetail](#)
- [ImportNameListData](#)
- [ModifyNameList](#)
- [ModifyNameListData](#)

新增数据结构：

- [DataContentInfo](#)
- [InputCreateNameListFront](#)
- [InputDeleteNameListData](#)
- [InputDeleteNameListFront](#)
- [InputDescribeDataListFront](#)
- [InputDescribeNameListDetail](#)
- [InputDescribeNameListFront](#)
- [InputImportNameListDataFront](#)
- [InputModifyNameFront](#)
- [InputModifyNameListDataFront](#)
- [InputModifyNameListDataFrontListData](#)
- [OutputDescribeDataListInfo](#)
- [OutputCreateNameListFront](#)
- [OutputDeleteNameListData](#)
- [OutputDeleteNameListFront](#)
- [OutputDescribeDataListFront](#)
- [OutputDescribeDataListFrontData](#)
- [OutputDescribeNameListDetail](#)
- [OutputDescribeNameListDetailFront](#)
- [OutputDescribeNameListFrontFix](#)
- [OutputDescribeNameListFrontFixListData](#)
- [OutputDescribeNameListInfo](#)
- [OutputImportNameListDataFront](#)
- [OutputModifyNameFront](#)
- [OutputModifyNameListFront](#)

## 第 11 次发布

发布时间：2024-07-23 01:57:03

本次发布包含了以下内容：

改善已有的文档。

新增数据结构：

- [DataAuthorizationInfo](#)

修改数据结构：

- [InputManageMarketingRisk](#)
  - 新增成员：DataAuthorization

## 第 10 次发布

发布时间：2024-05-17 02:07:15

本次发布包含了以下内容：

改善已有的文档。

删除接口：

- DescribeRiskAssessment

## 第 9 次发布

发布时间：2024-05-09 01:21:46

本次发布包含了以下内容：

改善已有的文档。

删除接口：

- DescribeRiskTrends

删除数据结构：

- InputFrontRisk
- OutputFrontRisk
- OutputFrontRiskData
- OutputFrontRiskValue

## 第 8 次发布

发布时间：2023-10-31 02:15:24

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- [OutputManageMarketingRiskValue](#)
  - 新增成员：RiskInformation

## 第 7 次发布

发布时间：2022-08-05 06:32:35

本次发布包含了以下内容：

改善已有的文档。

删除接口：

- DescribeRiskModel

删除数据结构：

- InputDescribeRiskModelData
- OutputDescribeRiskModel
- OutputDescribeRiskModelValue

## 第 6 次发布

发布时间：2022-04-21 06:16:10

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- [InputManageMarketingRisk](#)
  - 新增成员：Platform

- [OutputManageMarketingRiskValue](#)

- 新增成员: ConstId

## 第 5 次发布

发布时间: 2022-01-10 08:14:37

本次发布包含了以下内容:

改善已有的文档。

修改数据结构:

- InputDescribeRiskModelData
  - 新增成员: IsTest

## 第 4 次发布

发布时间: 2021-12-29 08:11:26

本次发布包含了以下内容:

改善已有的文档。

新增接口:

- DescribeRiskAssessment
- DescribeRiskTrends

新增数据结构:

- InputFrontRisk
- OutputFrontRisk
- OutputFrontRiskData
- OutputFrontRiskValue

## 第 3 次发布

发布时间: 2021-10-27 08:10:34

本次发布包含了以下内容:

改善已有的文档。

修改接口:

- [ManageMarketingRisk](#)
  - 新增入参: BusinessCryptoData
  - 修改入参: BusinessSecurityData

新增数据结构:

- [InputCryptoManageMarketingRisk](#)

## 第 2 次发布

发布时间: 2021-09-22 08:12:15

本次发布包含了以下内容:

改善已有的文档。

新增接口:

- DescribeRiskModel

新增数据结构:

- InputDescribeRiskModelData

- OutputDescribeRiskModel
- OutputDescribeRiskModelValue

## 第 1 次发布

发布时间：2021-01-27 08:10:11

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [ManageMarketingRisk](#)

新增数据结构：

- [AccountInfo](#)
- [InputDetails](#)
- [InputManageMarketingRisk](#)
- [OnlineScamInfo](#)
- [OtherAccountInfo](#)
- [OutputManageMarketingRisk](#)
- [OutputManageMarketingRiskValue](#)
- [QQAccountInfo](#)
- [SponsorInfo](#)
- [WeChatAccountInfo](#)



## 简介

最近更新时间：2021-06-23 11:40:05

欢迎使用 **全栈式风控引擎 API 3.0 版本**。全新的 API 接口文档更加规范和全面，统一的参数风格和公共错误码，统一的 SDK/CLI 版本与 API 文档严格一致，给您带来简单快捷的使用体验。支持全地域就近接入让您更快连接腾讯云产品。更多腾讯云 API 3.0 使用介绍请查看：[快速入门](#)

天御全栈式风控引擎，是基于人工智能技术和腾讯20年风控实战沉淀，依托腾讯海量业务构建的风控引擎，以轻量级的SaaS服务方式接入，帮助您快速解决注册、登录、营销活动等关键场景遇到的欺诈问题，实时防御黑灰产作恶

## API 概览

最近更新时间：2024-11-20 01:58:16

### 控制台相关接口

接口名称	接口功能	频率限制（次/秒）
CreateNameList	创建黑白名单	20
DeleteNameList	删除黑白名单	20
DescribeNameList	查询黑白名单	20
DescribeNameListDataList	查询黑白名单数据	20
DescribeUserUsageCnt	客户资源用量查询	20
ImportNameListData	添加黑白名单数据	20
ModifyNameList	修改黑白名单	20
ModifyNameListData	修改黑白名单数据	20
DeleteNameListData	删除黑白名单数据	20
DescribeNameListDetail	查询黑白名单详情	20

### API相关接口

接口名称	接口功能	频率限制（次/秒）
ManageMarketingRisk	全栈式风控引擎	1000

**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

最近更新時間：2025-03-28 14:27:43

### 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 `rce.tencentcloudapi.com`，也支持指定地域域名访问，例如广州地域的域名为 `rce.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 `rce.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）	<code>rce.tencentcloudapi.com</code>
华南地区(广州)	<code>rce.ap-guangzhou.tencentcloudapi.com</code>
华东地区(上海)	<code>rce.ap-shanghai.tencentcloudapi.com</code>
华北地区(北京)	<code>rce.ap-beijing.tencentcloudapi.com</code>
西南地区(成都)	<code>rce.ap-chengdu.tencentcloudapi.com</code>
西南地区(重庆)	<code>rce.ap-chongqing.tencentcloudapi.com</code>
港澳台地区(中国香港)	<code>rce.ap-hongkong.tencentcloudapi.com</code>
亚太东南(新加坡)	<code>rce.ap-singapore.tencentcloudapi.com</code>
亚太东南(曼谷)	<code>rce.ap-bangkok.tencentcloudapi.com</code>
亚太东北(首尔)	<code>rce.ap-seoul.tencentcloudapi.com</code>
亚太东北(东京)	<code>rce.ap-tokyo.tencentcloudapi.com</code>
美国东部(弗吉尼亚)	<code>rce.na-ashburn.tencentcloudapi.com</code>
美国西部(硅谷)	<code>rce.na-siliconvalley.tencentcloudapi.com</code>
欧洲地区(法兰克福)	<code>rce.eu-frankfurt.tencentcloudapi.com</code>

### 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 01:57:01

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。 <b>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。</b>
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。 <b>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。</b>
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输入公共参数 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 rce；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 <a href="#">文档</a> 。
Token	String	否	HTTP 请求头：X-TC-Token。即 <a href="#">安全凭证服务</a> 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。 <b>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。</b>
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 <a href="#">云API密钥</a> 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 <a href="#">文档</a> 。
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。

参数名称	类型	必选	描述
Token	String	否	即 <a href="#">安全凭证服务</a> 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=37ac2f4fde00b0ac9bd9eadeb459b1bbe224158d66e7ae5fcadb70b2d181d02&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=37ac2f4fde00b0ac9bd9eadeb459b1bbe224158d66e7ae5fcadb70b2d181d02&Region=ap-guangzhou&Nonce=23823223&SecretId=AKID*****
***
```

## 地域列表

本产品所有接口 Region 字段的可选值如下表所示。如果接口不支持该表中的所有地域，则会在接口文档中单独说明。

地域	取值
华南地区（广州）	ap-guangzhou

# 签名方法 v3

最近更新时间：2024-12-25 01:54:55

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。  此示例计算结果是 content-type:application/json; charset=utf-8\nhost:cvm.tencentcloudapi.com\nx-tc-action:describeinstances\n。 注意: content-type 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 charset 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。content-type 和 host 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 content-type;host;x-tc-action
HashedRequestPayload	请求正文 (payload, 即 body, 此示例为 {"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]} 的哈希值, 计算伪代码为 Lowercase(HexEncode(Hash.SHA256(RequestPayload))), 即对 HTTP 请求正文做 SHA256 哈希, 然后十六进制编

字段名称	解释
	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 35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064。

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 TC3-HMAC-SHA256。
RequestTimestamp	请求时间戳, 即请求头部的公共参数 X-TC-Timestamp 取值, 取当前时间 UNIX 时间戳, 精确到秒。此示例取值为 1551113065。
CredentialScope	凭证范围, 格式为 Date/service/tc3_request, 包含日期、所请求的服务和终止字符串 (tc3_request)。Date 为 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 为产品名, 必须与调用的产品域名一致。此示例计算结果是 2019-02-25/cvm/tc3_request。
HashedCanonicalRequest	前述步骤拼接所得规范请求串的哈希值, 计算伪代码为 Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))。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 是二进制的数，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=content-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=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f
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

```
#!/usr/bin/env 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": ["\u0672a\u0547d\u0540d"], "Name": "instance-name"}]}`
    hashedRequestPayload := sha256hex(payload)
    canonicalRequest := fmt.Sprintf("%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++)
{
    snprintf(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);

    #if 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 OPENSLL_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 OPENSLL_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%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。
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相符合，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

最近更新時間：2024-12-25 01:54:55

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
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。

将上一步排序好的请求参数格式化“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。**

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-gu
angzhou&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&R
egion=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 编码前需要以 UTF-8 进行编码。

**注意：**有些编程语言的网络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
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=7RAM2xfNMO9EiVTNmPg06MRnCvQ%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("evm.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 01:45:02

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 06:43:49

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 Golang 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。

# 控制台相关接口

## 创建黑白名单

最近更新时间：2025-03-20 12:10:49

### 1. 接口描述

接口请求域名：rce.tencentcloudapi.com。

创建黑白名单，黑白名单数量上限为100

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	<a href="#">公共参数</a> ，本接口取值：CreateNameList。
Version	是	String	<a href="#">公共参数</a> ，本接口取值：2020-11-03。
Region	是	String	<a href="#">公共参数</a> ，详见产品支持的 <a href="#">地域列表</a> 。
BusinessSecurityData	是	<a href="#">InputCreateNameListFront</a>	业务入参

### 3. 输出参数

参数名称	类型	描述
Data	<a href="#">OutputCreateNameListFront</a>	业务出参
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 CreateNameList

##### 输入示例

```
https://rce.tencentcloudapi.com/?Action=CreateNameList
&BusinessSecurityData.ListName="手机黑库"
&BusinessSecurityData.SceneCode="all_scene"
&BusinessSecurityData.Remark="测试"
&BusinessSecurityData.DataType=2
&BusinessSecurityData.ListType=2

<<公共请求参数>>
```

##### 输出示例

```
{
  "Response": {
```

```
"Data": {
  "Code": 0,
  "Message": "OK",
  "Value": []
},
"RequestId": "6ef60bec-0242-43af-bb20-270359Eb54a7"
}
```

## 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

该接口暂无业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

# 删除黑白名单

最近更新时间：2025-03-20 12:10:49

## 1. 接口描述

接口请求域名：rce.tencentcloudapi.com。

修改黑白名单状态 关闭 开启 删除

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	<a href="#">公共参数</a> ，本接口取值：DeleteNameList。
Version	是	String	<a href="#">公共参数</a> ，本接口取值：2020-11-03。
Region	是	String	<a href="#">公共参数</a> ，详见产品支持的 <a href="#">地域列表</a> 。
BusinessSecurityData	是	<a href="#">InputDeleteNameListFront</a>	业务入参

## 3. 输出参数

参数名称	类型	描述
Data	<a href="#">OutputDeleteNameListFront</a>	业务出参
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 DeleteNameList

#### 输入示例

```
https://rce.tencentcloudapi.com/?Action=DeleteNameList
&BusinessSecurityData.NameListId=33
&<公共请求参数>
```

#### 输出示例

```
{
  "Response": {
    "Data": {
      "Code": 0,
      "Message": "OK",
      "Value": []
    },
    "RequestId": "6ef60bec-0242-43af-bb20-270359Fb54a7"
  }
}
```

```
}  
}
```

## 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

该接口暂无业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

# 查询黑白名单

最近更新时间：2025-03-20 12:10:49

## 1. 接口描述

接口请求域名：rce.tencentcloudapi.com。

列表展示黑白名单列表数据，包含列表名称，名单类型，数据类型，数据来源，描述，状态等

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	<a href="#">公共参数</a> ，本接口取值：DescribeNameList。
Version	是	String	<a href="#">公共参数</a> ，本接口取值：2020-11-03。
Region	是	String	<a href="#">公共参数</a> ，详见产品支持的 <a href="#">地域列表</a> 。
BusinessSecurityData	是	<a href="#">InputDescribeNameListFront</a>	业务入参

## 3. 输出参数

参数名称	类型	描述
Data	<a href="#">OutputDescribeNameListFrontFixListData</a>	业务出参
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 DescribeNameList

#### 输入示例

```
https://rce.tencentcloudapi.com/?Action=DescribeNameList

&BusinessSecurityData.Status=1
&BusinessSecurityData.DataType=1
&BusinessSecurityData.ListType=1
&BusinessSecurityData.PageNumber=1
&BusinessSecurityData.PageSize=10
&<公共请求参数>
```

#### 输出示例

```
{
  "Response": {
    "Data": {},
    "RequestId": "6ef60bec-0242-43af-bb20-270359fb54a7"
  }
}
```



```
}  
}
```

## 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

该接口暂无业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

# 查询黑白名单数据

最近更新时间：2025-03-20 12:10:49

## 1. 接口描述

接口请求域名：rce.tencentcloudapi.com。

黑白名单详情数据展示 名单id 客户appid uin 数据内容 开始时间和结束时间 状态 描述

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	<a href="#">公共参数</a> ，本接口取值：DescribeNameListDataList。
Version	是	String	<a href="#">公共参数</a> ，本接口取值：2020-11-03。
Region	是	String	<a href="#">公共参数</a> ，详见产品支持的 <a href="#">地域列表</a> 。
BusinessSecurityData	是	<a href="#">InputDescribeDataListFront</a>	业务入参

## 3. 输出参数

参数名称	类型	描述
Data	<a href="#">OutputDescribeDataListFrontData</a>	业务出参
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 DescribeNameListDataList

#### 输入示例

```
https://rce.tencentcloudapi.com/?Action=DescribeNameListDataList
&BusinessSecurityData.NameListId=33
&BusinessSecurityData.Status=1
&BusinessSecurityData.PageNumber=1
&BusinessSecurityData.PageSize=10
&<公共请求参数>
```

#### 输出示例

```
{
  "Response": {
    "Data": {},
    "RequestId": "6ef60bec-0242-43af-bb20-270359fb54a7"
  }
}
```

## 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

该接口暂无业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

# 客户资源用量查询

最近更新时间：2025-03-20 12:10:48

## 1. 接口描述

接口请求域名：rce.tencentcloudapi.com。

RCE控制台预付费和后付费次数展示

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	<a href="#">公共参数</a> ，本接口取值：DescribeUserUsageCnt。
Version	是	String	<a href="#">公共参数</a> ，本接口取值：2020-11-03。
Region	是	String	<a href="#">公共参数</a> ，详见产品支持的 <a href="#">地域列表</a> 。

## 3. 输出参数

参数名称	类型	描述
Data	<a href="#">OutputDescribeUserUsageCntData</a>	业务出参
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 demo数据

#### 输入示例

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

{}
```

#### 输出示例

```
{
  "Response": {
    "Data": {
      "Code": 0,
      "Message": "OK",
      "Value": {
        "PayMode": 0,
```

```
"AfterPayModeThisMonthUsedCnt": 0,
"CreateTime": "2024-11-01 00:00:00",
"ExpireTime": "2025-11-01 00:00:00",
"AfterPayModeLastMonthUsedCnt": 0,
"BeforePayModeTotalUsedCnt": 0,
"BeforePayModeRemainUsedCnt": 0
},
"RequestId": "6ef60bec*****270359fb54a7"
}
```

## 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

该接口暂无业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

# 添加黑白名单数据

最近更新时间：2025-03-20 12:10:48

## 1. 接口描述

接口请求域名：rce.tencentcloudapi.com。

新增黑白名单数据，所有黑白名单数据总量上限为10000

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	<a href="#">公共参数</a> ，本接口取值：ImportNameListData。
Version	是	String	<a href="#">公共参数</a> ，本接口取值：2020-11-03。
Region	是	String	<a href="#">公共参数</a> ，详见产品支持的 <a href="#">地域列表</a> 。
BusinessSecurityData	是	<a href="#">InputImportNameListDataFront</a>	业务入参

## 3. 输出参数

参数名称	类型	描述
Data	<a href="#">OutputImportNameListDataFront</a>	业务出参
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 ImportNameListData

#### 输入示例

```
https://rce.tencentcloudapi.com/?Action=ImportNameListData
&BusinessSecurityData.NameListId=33
&BusinessSecurityData.DataSource=2
&BusinessSecurityData.DataContentInfo.0.DataContent="xxx.xx.0.1";
&<公共请求参数>
```

#### 输出示例

```
{
  "Response": {
    "Data": {
      "Message": "OK",
      "Code": 0,
      "Value": []
    }
  }
}
```

```
"RequestId": "6ef60bec-0242-43af-bb20-270359fb54a7"  
}  
}
```

## 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

该接口暂无业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

# 修改黑白名单

最近更新时间：2025-03-20 12:10:48

## 1. 接口描述

接口请求域名：rce.tencentcloudapi.com。

修改列表数据 列表名称 列表类型 数据类型 状态 备注

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	<a href="#">公共参数</a> ，本接口取值：ModifyNameList。
Version	是	String	<a href="#">公共参数</a> ，本接口取值：2020-11-03。
Region	是	String	<a href="#">公共参数</a> ，详见产品支持的 <a href="#">地域列表</a> 。
BusinessSecurityData	是	<a href="#">InputModifyNameFront</a>	业务入参

## 3. 输出参数

参数名称	类型	描述
Data	<a href="#">OutputModifyNameFront</a>	业务出参
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 ModifyNameList

#### 输入示例

```
https://rce.tencentcloudapi.com/?Action=ModifyNameList
&BusinessSecurityData.NameListId=33
&<公共请求参数>
```

#### 输出示例

```
{
  "Response": {
    "Data": {
      "Message": "OK",
      "Code": 0,
      "Value": []
    },
    "RequestId": "6ef60bec-0242-43af-bb20-270359Fb54a7"
  }
}
```



```
}  
}
```

## 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

该接口暂无业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

# 修改黑白名单数据

最近更新时间：2025-03-20 12:10:48

## 1. 接口描述

接口请求域名：rce.tencentcloudapi.com。

修改黑白名单列表详情 详情内容 开始和结束时间 状态 备注等

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	<a href="#">公共参数</a> ，本接口取值：ModifyNameListData。
Version	是	String	<a href="#">公共参数</a> ，本接口取值：2020-11-03。
Region	是	String	<a href="#">公共参数</a> ，详见产品支持的 <a href="#">地域列表</a> 。
BusinessSecurityData	是	<a href="#">InputModifyNameListDataFrontListData</a>	业务入参

## 3. 输出参数

参数名称	类型	描述
Data	<a href="#">OutputModifyNameListFront</a>	业务出参
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 ModifyNameListData

#### 输入示例

```
https://rce.tencentcloudapi.com/?Action=ModifyNameListData
&BusinessSecurityData.DataList.0.Status=1
&BusinessSecurityData.DataList.0.Remark="测试"
&BusinessSecurityData.DataList.0.DataContent="xxx.xx.12.123"
&BusinessSecurityData.DataList.0.NameListDataId=73
&BusinessSecurityData.DataList.0.StartTime="2020-03-02 20:50:00"
&BusinessSecurityData.DataList.0.EndTime="2020-03-02 20:50:00"
&<公共请求参数>
```

#### 输出示例

```
{
  "Response": {
    "Data": {
      "Code": 0,
```

```
"Message": "OK",
"Value": []
},
"RequestId": "6ef60bec-0242-43af-bb20-270359fb54a7"
}
```

## 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

该接口暂无业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

# 查询黑白名单详情

最近更新时间：2025-03-20 12:10:48

## 1. 接口描述

接口请求域名：rce.tencentcloudapi.com。

查询黑白名单列表详情

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	<a href="#">公共参数</a> ，本接口取值：DescribeNameListDetail。
Version	是	String	<a href="#">公共参数</a> ，本接口取值：2020-11-03。
Region	是	String	<a href="#">公共参数</a> ，详见产品支持的 <a href="#">地域列表</a> 。
BusinessSecurityData	否	<a href="#">InputDescribeNameListDetail</a>	业务入参

## 3. 输出参数

参数名称	类型	描述
Data	<a href="#">OutputDescribeNameListDetailFront</a>	黑白名单列表详情业务出参
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 查询黑白名单详情

#### 输入示例

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

{
  "BusinessSecurityData": {
    "NameListId": 2446
  }
}
```

#### 输出示例

```
{
  "Response": {
    "Data": {
      "Code": 0,
      "Message": "OK",
      "Value": {
        "NameListId": 2446,
        "ListName": "手机号黑名单",
        "ListType": 1,
        "DataType": 4,
        "SceneCode": "all_scene",
        "Status": 1,
        "Remark": "手机号黑名单",
        "CreateTime": "2024-05-22 14:58:21",
        "UpdateTime": "2024-07-15 14:29:35",
        "EncryptionType": 0
      }
    },
    "RequestId": "bd2f3d71-9102-4c24-b581-4d4f22ebd077"
  }
}
```

## 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

该接口暂无业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

# 删除黑白名单数据

最近更新时间：2025-03-20 12:10:49

## 1. 接口描述

接口请求域名：rce.tencentcloudapi.com。

删除黑白名单数据

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：DeleteNameListData。
Version	是	String	公共参数，本接口取值：2020-11-03。
Region	是	String	公共参数，详见产品支持的 <a href="#">地域列表</a> 。
BusinessSecurityData	否	InputDeleteNameListData	业务入参

## 3. 输出参数

参数名称	类型	描述
Data	OutputDeleteNameListData	业务出参
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 删除名单数据

#### 输入示例

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

{
  "BusinessSecurityData": {
    "NameListDataIdList": [
      177773
    ]
  }
}
```

#### 输出示例

```
{
  "Response": {
    "Data": {
      "Code": 0,
      "Message": "OK",
      "Value": null
    },
    "RequestId": "8bf7ba3a-80cd-4d24-af5e-e02acee93539"
  }
}
```

## 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

该接口暂无业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

# API相关接口

## 全栈式风控引擎

最近更新時間：2025-03-28 14:27:43

### 1. 接口描述

接口請求域名：rce.tencentcloudapi.com。

全栈式风控引擎（RiskControlEngine，RCE）是基于人工智能技术和腾讯20年风控实战沉淀，依托腾讯海量业务构建的风控引擎，以轻量级的 SaaS 服务方式接入，帮助您快速解决注册、登录、营销活动等关键场景遇到的欺诈问题，实时防御黑灰产作恶。

默认接口请求频率限制：1000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	<a href="#">公共参数</a> ，本接口取值：ManageMarketingRisk。
Version	是	String	<a href="#">公共参数</a> ，本接口取值：2020-11-03。
Region	是	String	<a href="#">公共参数</a> ，详见产品支持的 <a href="#">地域列表</a> 。
BusinessSecurityData	否	<a href="#">InputManageMarketingRisk</a>	业务入参
BusinessCryptoData	否	<a href="#">InputCryptoManageMarketingRisk</a>	业务入参

### 3. 输出参数

参数名称	类型	描述
Data	<a href="#">OutputManageMarketingRisk</a>	业务出参
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 ManageMarketingRisk

使用手机号MD5请求方式

输入示例

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

{
  "BusinessSecurityData": {
    "Account": {
      "AccountType": 10004,
```



```
"OtherAccount": {
  "AccountId": "3ac9aa*****526ed9"
},
"SceneCode": "e_activity_antirush",
"UserIp": "113.***.***.150",
"PostTime": 1712841851,
"DataAuthorization": {
  "DataProviderName": "某某有限公司",
  "DataRecipientName": "腾讯云计算（北京）有限责任公司",
  "UserDataTypes": [
    1,
    4
  ],
  "IsAuthorize": 1,
  "IsOrderHandling": 1,
  "AuthorizationTerm": 1728714893,
  "PrivacyPolicyLink": "https://www.*****.com/"
}
}
```

#### 输出示例

```
{
  "Response": {
    "Data": {
      "Code": 0,
      "Message": "OK",
      "Uuid": "9a570741-****-****-****-****0b788232",
      "Value": {
        "AssociateAccount": "",
        "ConstId": "",
        "PostTime": 1712841851,
        "RiskInformation": null,
        "RiskLevel": "reject",
        "RiskType": [
          201,
          1,
          21,
          11,
          2011
        ],
        "UserId": "3ac9aa*****526ed9",
        "UserIp": "113.***.***.150"
      }
    },
    "RequestId": "5d9e8c5d-****-****-****-1c0dca666edb"
  }
}
```

## 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

该接口暂无业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

## 数据结构

最近更新时间：2025-03-28 14:27:43

### AccountInfo

账号信息。

被如下接口引用：ManageMarketingRisk。

名称	类型	必选	描述
AccountType	Integer	是	用户账号类型；默认开通QQOpenId、手机号MD5权限；如果需要使用微信OpenId入参，则需要“提交工单”或联系对接人进行资格审核，审核通过后方可正常使用微信开放账号。 1: QQ开放账号 2: 微信开放账号 10004: 手机号MD5，中国大陆11位手机号进行MD5加密，取32位小写值 10005: 手机号SHA256，中国大陆11位手机号进行SHA256加密，取64位小写值 示例值：10004
QQAccount	<a href="#">QQAccountInfo</a>	否	QQ账号信息，AccountType是"1"时，该字段必填。 示例值：QQAccount.QQOpenId="A8E0232CD*****2058B0EA885"
WeChatAccount	<a href="#">WeChatAccountInfo</a>	否	微信账号信息，AccountType是"2"时，该字段必填。 示例值：WeChatAccount.WeChatOpenId="oOya25F*****4OQCdcFo"
OtherAccount	<a href="#">OtherAccountInfo</a>	否	其它账号信息，AccountType是10004或10005时，该字段必填。 示例值：3ac9aa8a9a0074918763bfd6ed526ed9

### DataAuthorizationInfo

数据授权信息

被如下接口引用：ManageMarketingRisk。

名称	类型	必选	描述
DataProviderName	String	是	数据委托方、需求方：客户主体名称。 示例值：某某有限公司
DataRecipientName	String	是	数据受托方、提供方：腾讯云主体名称。固定填：腾讯云计算（北京）有限责任公司 示例值：腾讯云计算（北京）有限责任公司
UserDataType	Array of Integer	是	客户请求RCE所提供的用户数据类型，支持多选。实际以接口请求传参为准。 1-手机号； 2-微信开放账号； 3-QQ开放账号； 4-IP地址； 999-其它； 示例值：[1, 4]
IsAuthorize	Integer	是	客户是否已按 <a href="#">合规指南</a> 要求获取用户授权，同意客户委托腾讯云处理入参信息 1-已授权；其它值为未授权。 示例值：1
IsOrderHandling	Integer	否	客户是否已按 <a href="#">合规指南</a> 要求获取用户授权，同意腾讯云结合客户提供的信息，对已合法收集的用户数据进行必要处理得出服务结果，并返回给客户。 1-已授权；其它值为未授权。 示例值：1
AuthorizationTerm	Integer	否	客户获得的用户授权期限时间戳（单位秒）。不填默认无固定期限。 示例值：1719805604
PrivacyPolicyLink	String	否	客户获得用户授权所依赖的协议地址。 示例值：https://www.\*\*\*.com/\*

### DataContentInfo

黑白名单导入名单数据的业务入参数据结构

被如下接口引用: ImportNameListData。

名称	类型	必选	描述
DataContent	String	否	名单数据内容 示例值: 3ac9aa*****526ed9
DataRemark	String	否	名单数据描述 示例值: 手机号
StartTime	String	否	名单数据开始时间 示例值: 2025-02-13 12:11:12
EndTime	String	否	名单数据结束时间 示例值: 2025-02-13 12:11:12

## InputCreateNameListFront

创建黑白名单入参

被如下接口引用: CreateNameList。

名称	类型	必选	描述
ListName	String	是	名单名称 示例值: 手机号黑名单
ListType	Integer	是	名单类型 [1 黑名单 2 白名单] 示例值: 1
DataType	Integer	是	数据类型 [1 手机号 2 qqOpenId 3 wechatOpenId 4 ip 6 idfa 7 imei] 示例值: 1
Remark	String	否	描述 示例值: 手机号名单
EncryptionType	Integer	否	加密类型 [0 无需加密 1 MD5加密 2 SHA256加密] 示例值: 0
SceneCode	String	否	场景Code, all_scene代表全部场景 示例值: e_activity_antirush

## InputCryptoManageMarketingRisk

全栈式风控引擎入参

被如下接口引用: ManageMarketingRisk。

名称	类型	必选	描述
IsAuthorized	String	否	是否授权: 1已授权, 否则未授权。 调用全栈式风控引擎接口服务时, 客户需先明确授权。 示例值: 1
CryptoType	String	否	加密类型: 1AES加密 示例值: 1
CryptoContent	String	否	加密内容, 非空时接口采用加密模式。 示例值: tLMs/D/1SHZD/*****OoqdlZRQk8mmQJTUiY3xLw=

## InputDeleteNameListData

删除黑白名单数据业务入参

被如下接口引用: DeleteNameListData。

名称	类型	必选	描述
NameListDataIdList	Array of Integer	是	黑白名单数据ID集合 示例值: 1001

## InputDeleteNameListFront

删除黑白名单入参

被如下接口引用: DeleteNameList。

名称	类型	必选	描述
NameListId	Integer	是	名单ID 示例值: 1001

## InputDescribeDataListFront

查询黑白名单数据入参

被如下接口引用: DescribeNameListDataList。

名称	类型	必选	描述
NameListId	Integer	是	名单ID 示例值: 1001
PageNumber	Integer	是	当前页数 示例值: 1
PageSize	Integer	是	每页显示条数 示例值: 50
KeyWord	String	否	搜索关键字, 按照名单数据名称或加密名单数据名称搜索 示例值: 3ac9aa*****526ed9
Status	Integer	否	黑白名单列表状态[1 启用 2 停用] 示例值: 1

## InputDescribeNameListDetail

查询黑白名单详情入参

被如下接口引用: DescribeNameListDetail。

名称	类型	必选	描述
NameListId	Integer	是	名单ID 示例值: 1001

## InputDescribeNameListFront

查询黑白名单入参

被如下接口引用: DescribeNameList。

名称	类型	必选	描述
PageNumber	Integer	是	当前页数 示例值: 1
PageSize	Integer	是	每页显示条数 示例值: 50
ListType	Integer	否	名单类型 [1 黑名单 2 白名单] 示例值: 1

名称	类型	必选	描述
DataType	Integer	否	数据类型[1 手机号 2 qqOpenId 3 wechatOpenId 4 ip 6 idfa 7 imei] 示例值: 1
KeyWord	String	否	关键字, 按照名单名称搜索 示例值: 3ac9aa*****526ed9
Status	Integer	否	记录状态[1 启用 2 停用] 示例值: 1

## InputDetails

入参的详细参数信息

被如下接口引用: ManageMarketingRisk。

名称	类型	必选	描述
FieldName	String	是	字段名称 示例值: openBuyTime
FieldValue	String	是	字段值 示例值: 1739268000000

## InputImportNameListDataFront

添加名单数据入参

被如下接口引用: ImportNameListData。

名称	类型	必选	描述
NameListId	Integer	是	名单ID 示例值: 1001
DataSource	Integer	是	数据来源, 固定传2 (手工录入) 示例值: 2
DataContentInfo	Array of <a href="#">DataContentInfo</a>	否	黑白名单数据内容

## InputManageMarketingRisk

全栈式风控引擎入参

被如下接口引用: ManageMarketingRisk。

名称	类型	必选	描述
Account	<a href="#">AccountInfo</a>	是	用户账号类型; 默认开通QQOpenId、手机号MD5权限; 如果需要使用微信OpenId入参, 则需要"提交工单"或联系对接人进行资格审核, 审核通过后方可正常使用微信开放账号。 1: QQ开放账号 2: 微信开放账号 10004: 手机号MD5, 中国大陆11位手机号进行MD5加密, 取32位小写值 10005: 手机号SHA256, 中国大陆11位手机号进行SHA256加密, 取64位小写值 示例值: AccountInfo.AccountType= 10004
SceneCode	String	是	场景码, 用于识别和区分不同的业务场景, 可在控制台上新建和管理 控制台链接: <a href="https://console.cloud.tencent.com/rce/risk/strategy/scene-root">https://console.cloud.tencent.com/rce/risk/strategy/scene-root</a> 活动防刷默认场景码: e_activity_antirush 登录保护默认场景码: e_login_protection 注册保护默认场景码: e_register_protection 示例值: e_activity_antirush
UserIp	String	是	用户外网ip (传入用户非外网ip会影响判断结果)。 示例值: 123.***.***.253

名称	类型	必选	描述
PostTime	Integer	是	用户操作时间戳，精确到秒。 示例值：1615211954
UserId	String	否	业务平台用户唯一标识，支持自定义。 示例值：100001
DeviceToken	String	否	设备指纹DeviceToken值，集成设备指纹后获取；如果集成了相应的设备指纹，该字段必填。 示例值：v2:DJsBrn.....(此处省略若干字符)jJNkBydl=
DeviceBusinessId	Integer	否	设备指纹 BusinessId。 示例值：DE****123
BusinessId	Integer	否	业务ID。网站或应用在多个业务中使用此服务，通过此ID区分统计数据。 示例值：1001
Nickname	String	否	昵称，UTF-8 编码。 示例值：zhangsan
EmailAddress	String	否	用户邮箱地址。 示例值：276****396@qq.com
CheckDevice	Integer	否	是否识别设备异常： 0：不识别。 1：识别。 示例值：1
CookieHash	String	否	用户HTTP请求中的Cookie进行2次hash的值，只要保证相同Cookie的hash值一致即可。 示例值：fa5805903fb6b1f7e36bb1c2fd59e47684b8d180
Referer	String	否	用户HTTP请求的Referer值。 示例值：fa5805903fb6b1f7e36bb1c2fd59e47684b8d180
UserAgent	String	否	用户HTTP请求的User-Agent值。 示例值：Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
XForwardedFor	String	否	用户HTTP请求的X-Forwarded-For值。 示例值：192.***.***.1
MacAddress	String	否	MAC地址或设备唯一标识。 示例值：00:1A:2B:3C:4D:5E
VendorId	String	否	手机制造商ID，如果手机注册，请带上此信息。 示例值：huaweiMate10pro
DeviceType	Integer	否	设备类型(已不推荐使用)。 示例值：1
Details	Array of <a href="#">InputDetails</a>	否	扩展字段。 示例值：Details[0].FieldName='system_version' Details[0].FieldValue='V1.5'
Sponsor	<a href="#">SponsorInfo</a>	否	邀请助力场景相关信息。 示例值：Sponsor.SponsorOpenId='otg-dasndasjdasldsalkjdlksad'
OnlineScam	<a href="#">OnlineScamInfo</a>	否	详情请跳转至 <a href="#">OnlineScamInfo</a> 查看。 示例值：OnlineScam.ContentLabel = '诈骗'
Platform	String	否	1: Android 2: iOS 3: H5 4: 小程序 示例值：1
DataAuthorization	<a href="#">DataAuthorizationInfo</a>	否	数据授权信息。

## InputModifyNameFront

修改黑白名单入参

被如下接口引用：[ModifyNameList](#)。

名称	类型	必选	描述
NameListId	Integer	是	名单ID 示例值: 1001
ListName	String	否	名单名称 示例值: 手机号黑名单
Status	Integer	否	名单状态 [1 启用 2 停用] 示例值: 1
Remark	String	否	描述 示例值: 手机号黑名单

## InputModifyNameListDataFront

名单数据集合

被如下接口引用: ModifyNameListData。

名称	类型	必选	描述
NameListDataId	Integer	是	名单数据ID 示例值: 10010
DataContent	String	否	名单数据内容 示例值: 3ac9aa*****526ed9
StartTime	String	否	名单数据开始时间 示例值: 2024-05-05 12:10:15
EndTime	String	否	名单数据结束时间 示例值: 2024-05-05 12:10:15
Status	Integer	否	记录状态 [1 启用 2 停用] 示例值: 1
Remark	String	否	名单数据描述 示例值: 手机号黑名单

## InputModifyNameListDataFrontListData

修改黑白名单数据入参

被如下接口引用: ModifyNameListData。

名称	类型	必选	描述
DataList	Array of <a href="#">InputModifyNameListDataFront</a>	否	名单数据集合

## OnlineScamInfo

诈骗信息。

被如下接口引用: ManageMarketingRisk。

名称	类型	必选	描述
ContentLabel	String	否	内容标签。 示例值: 诈骗
ContentRiskLevel	Integer	否	内容风险等级: 0: 正常。 1: 可疑。 示例值: 0
ContentType	Integer	否	内容产生形式: 0: 对话。



名称	类型	必选	描述
			1: 广播。 示例值: 0
FraudType	Integer	否	账号类型 1: 手机号 2: uin账号 示例值: 1
FraudAccount	String	否	账号 示例值: 3ac9aa*****526ed9

## OtherAccountInfo

其它账号信息。

被如下接口引用: ManageMarketingRisk。

名称	类型	必选	描述
AccountId	String	是	其他账号信息; AccountType是10004时, 填入中国大陆标准11位手机号的MD5值 AccountType是10005时, 填入中国大陆标准11位手机号的SHA256值 注释: MD5手机号加密方式, 使用中国大陆11位手机号进行MD5加密, 加密后取32位小写值。 SHA256手机号加密方式, 使用中国大陆11位手机号进行SHA256加密, 加密后取64位小写值。 示例值: 3ac9aa*****526ed9
MobilePhone	String	否	账号绑定的MD5或SHA256加密的手机号(该字段已不推荐使用)。 注释: 支持标准中国大陆11位手机号MD5加密后位的32位小写字符串; 支持标准中国大陆11位手机号SHA256加密后位的64位小写字符串。 示例值: 3ac9aa*****526ed9
DeviceId	String	否	用户设备号(该字段已不推荐使用)。 示例值: 095006*****dddc39

## OutputDescribeDataListInfo

黑白名单数据列表信息

被如下接口引用: DescribeNameListDataList。

名称	类型	描述
Count	Integer	数量 示例值: 100
List	Array of <a href="#">OutputDescribeDataListFront</a>	列表 注意: 此字段可能返回 null, 表示取不到有效值。

## OutputCreateNameListFront

创建黑白名单出参

被如下接口引用: CreateNameList。

名称	类型	描述
Code	Integer	错误码, 0 表示成功, 非0表示失败错误码。0: 成功 1002: 参数错误 4300: 未开通服务 6000: 系统内部错误 示例值: 0
Message	String	错误信息 示例值: OK
Value	Array of String	空数组 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: []

## OutputDeleteNameListData

删除黑白名单出参

被如下接口引用: DeleteNameListData。

名称	类型	描述
Code	Integer	错误码, 0 表示成功, 非0表示失败错误码。0: 成功 1002: 参数错误 4300: 未开通服务 6000: 系统内部错误 示例值: 0
Message	String	错误信息 示例值: OK
Value	Array of String	空数组 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: []

## OutputDeleteNameListFront

删除黑白名单出参

被如下接口引用: DeleteNameList。

名称	类型	描述
Code	Integer	错误码, 0 表示成功, 非0表示失败错误码。0: 成功 1002: 参数错误 4300: 未开通服务 6000: 系统内部错误 示例值: 0
Message	String	错误信息 示例值: OK
Value	Array of String	空数组 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: []

## OutputDescribeDataListFront

黑白名单数据信息

被如下接口引用: DescribeNameListDataList。

名称	类型	描述
NameListDataId	Integer	名单数据ID 示例值: 10010
NameListId	Integer	名单ID 示例值: 1001
DataContent	String	名单数据内容 示例值: 137***98
DataSource	Integer	数据来源, 固定传2 (手工录入) 示例值: 2
StartTime	String	名单数据开始时间 示例值: 2024-05-05 12:10:15
EndTime	String	名单数据结束时间 示例值: 2024-05-05 12:10:15
Status	Integer	名单数据状态 [1 启用 2 停用] 示例值: 1
Remark	String	名单数据描述 示例值: 手机号黑名单
CreateTime	String	名单数据创建时间 示例值: 2024-05-05 12:10:15

名称	类型	描述
UpdateTime	String	名单数据更新时间 示例值: 2024-05-05 12:10:15
EncryptDataContent	String	加密名单数据内容 示例值: aba***md

## OutputDescribeDataListFrontData

查询黑白名单数据出参

被如下接口引用: DescribeNameListDataList。

名称	类型	描述
Code	Integer	错误码, 0 表示成功, 非0表示失败错误码。0: 成功 1002: 参数错误 4300: 未开通服务 6000: 系统内部错误 示例值: 0
Message	String	错误信息 示例值: OK
Value	<a href="#">OuntputDescribeDataListInfo</a>	黑白名单数据信息

## OutputDescribeNameListDetail

黑白名单详情出参

被如下接口引用: DescribeNameListDetail。

名称	类型	描述
NameListId	Integer	名单ID 示例值: 1001
ListName	String	名单名称 示例值: 手机号黑名单
ListType	Integer	名单类型 [1 黑名单 2 白名单] 示例值: 1
DataType	Integer	数据类型[1 手机号 2 qqOpenId 3 wechatOpenId 4 ip 6 idfa 7 imei] 示例值: 1
SceneCode	String	场景Code 示例值: e_activity_antirush
Status	Integer	名单列表状态 [1 启用 2 停用] 示例值: 1
Remark	String	描述 示例值: 手机号黑名单
CreateTime	String	创建时间 示例值: 2024-05-05 12:10:15
UpdateTime	String	更新时间 示例值: 2024-05-05 12:10:15
EncryptionType	Integer	加密类型 [0 无需加密, 1 MD5加密, 2 SHA256加密] 示例值: 0

## OutputDescribeNameListDetailFront

查询列表详情出参

被如下接口引用: DescribeNameListDetail。

名称	类型	描述
Code	Integer	错误码, 0 表示成功, 非0表示失败错误码。0: 成功 1002: 参数错误 4300: 未开通服务 6000: 系统内部错误 示例值: 0
Message	String	错误信息 示例值: OK
Value	<a href="#">OutputDescribeNameListDetail</a>	列表详情信息 注意: 此字段可能返回 null, 表示取不到有效值。

## OutputDescribeNameListFrontFix

黑白名单信息

被如下接口引用: DescribeNameList。

名称	类型	描述
NameListId	Integer	名单ID 示例值: 1001
ListName	String	名单名称 示例值: 手机号黑名单
ListType	Integer	名单类型 [1 黑名单 2 白名单] 示例值: 1
DataType	Integer	数据类型[1 手机号 2 qqOpenId 3 wechatOpenId 4 ip 6 idfa 7 imei] 示例值: 1
Status	Integer	记录状态 [1 启用 2 停用] 示例值: 1
Remark	String	描述 示例值: 手机号黑名单
CreateTime	String	创建时间 示例值: 2024-05-05 12:10:15
UpdateTime	String	更新时间 示例值: 2024-05-05 12:10:15
EffectCount	String	有效数据/数据总数 示例值: 10/20
EncryptionType	Integer	加密类型[0 无需加密 1 MD5加密 2 SHA256加密] 示例值: 0
SceneCode	String	场景Code, all_scene代表全部场景 示例值: all_scene

## OutputDescribeNameListFrontFixListData

查询黑白名单出参

被如下接口引用: DescribeNameList。

名称	类型	描述
Code	Integer	错误码, 0 表示成功, 非0表示失败错误码。0: 成功 1002: 参数错误 4300: 未开通服务 6000: 系统内部错误。 示例值: 0
Message	String	错误信息。 示例值: OK
Value	<a href="#">OutputDescribeNameListInfo</a>	黑白名单列表信息。

## OutputDescribeNameListInfo

黑白名单信息

被如下接口引用: DescribeNameList。

名称	类型	描述
Count	Integer	总数 示例值: 100
List	Array of <a href="#">OutputDescribeNameListFrontFix</a>	列表 注意: 此字段可能返回 null, 表示取不到有效值。

## OutputDescribeUserUsageCnt

预付费 后付费 总数显示接口出参

被如下接口引用: DescribeUserUsageCnt。

名称	类型	描述
PayMode	Integer	当前付费模式, 0 后付费 1 预付费 示例值: 0
AfterPayModeThisMonthUsedCnt	Integer	后付费本月使用量 示例值: 0
CreateTime	String	创建时间 示例值: 2000-01-01 00:00:00
ExpireTime	String	超出时间 示例值: 2000-01-01 00:00:00
AfterPayModeLastMonthUsedCnt	Integer	后付费上月使用量 示例值: 0
BeforePayModeTotalUsedCnt	Integer	预付费总量 示例值: 0
BeforePayModeRemainUsedCnt	Integer	预付费剩余用量 示例值: 0

## OutputDescribeUserUsageCntData

RCE控制台预付费和后付费次数展示参数数据

被如下接口引用: DescribeUserUsageCnt。

名称	类型	描述
Code	Integer	错误码, 0 表示成功, 非0表示失败错误码。0: 成功 4300: 未开通服务 示例值: 0
Message	String	错误信息 示例值: OK
Value	<a href="#">OutputDescribeUserUsageCnt</a>	业务详情 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: 0

## OutputImportNameListDataFront

添加黑白名单数据出参

被如下接口引用: ImportNameListData。

名称	类型	描述
Code	Integer	错误码, 0 表示成功, 非0表示失败错误码。0: 成功 1002: 参数错误 4300: 未开通服务 6000: 系统内部错误 示例值: 0
Message	String	错误信息 示例值: OK
Value	Array of String	空数组 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: []

## OutputManageMarketingRisk

全栈式风控引擎出参

被如下接口引用: ManageMarketingRisk。

名称	类型	描述
Code	Integer	错误码, 0 表示成功, 非0表示失败错误码。 0: 成功 1: 错误 1002: 参数错误 4300: 未开通服务 4301: 后端未创建对应产品 示例值: 0
Message	String	UTF-8编码, 出错消息。 示例值: OK
Value	<a href="#">OutputManageMarketingRiskValue</a>	业务详情。 注意: 此字段可能返回 null, 表示取不到有效值。
UUid	String	控制台显示的req_id。 示例值: 9a570741-****-****-****-****0b788232

## OutputManageMarketingRiskValue

全栈式风控引擎出参值

被如下接口引用: ManageMarketingRisk。

名称	类型	描述
UserId	String	账号ID: 对应输入参数。 当AccountType为1时, 对应QQ的OpenId; 当AccountType为2时, 对应微信的OpenId/UnionId; 当AccountType为10004时, 对应手机号的MD5值; 当AccountType为10005时, 对应手机号的SHA256值。 示例值: 3ac9aa8a9a0*****fd6ed526ed9
PostTime	Integer	操作时间戳, 单位秒 (对应输入参数)。 示例值: 1614760549
AssociateAccount	String	业务预留参数 (暂无实际业务含义, 无需关注返回值) 示例值: 1***01
UserIp	String	操作来源的外网IP (对应输入参数)。 示例值: 123.***.***.253
RiskLevel	String	风险等级 pass: 无恶意 review: 低风险, 需要人工审核 reject: 高风险, 建议拦截 示例值: reject
RiskType	Array of Integer	风险类型, 可能同时命中多个风险类型 1: 账号信用低, 账号近期存在因恶意被处罚历史, 网络低活跃, 被举报等因素。

名称	类型	描述
		11: 疑似低活跃账号, 账号活跃度与正常用户有差异。 2: 垃圾账号, 疑似批量注册小号, 近期存在严重违规或大量举报。 21: 疑似小号, 账号有疑似线上养号, 小号等行为。 22: 疑似违规账号, 账号曾有违规行为、曾被举报过、曾因违规被处罚过等。 3: 无效账号, 送检账号参数无法成功解析, 请检查微信 OpenId 是否有误/AppId 与 QQ OpenId 无法关联/微信 OpenId 权限是否开通/手机号是否为中国大陆手机号; 4: 黑名单, 该账号在业务侧有过拉黑记录。 5: 白名单, 业务自行有添加过白名单记录。 101: 批量操作, 存在 IP/设备/环境等因素的聚集性异常。 1011: 疑似 IP 属性聚集, 出现 IP 聚集。 1012: 疑似设备属性聚集, 出现设备聚集。 102: 自动机, 疑似自动机批量请求。 103: 恶意行为-网赚, 疑似网赚。 104: 微信登录态无效, 检查 WeChatAccessToken 参数, 是否已经失效。 201: 环境风险, 环境异常操作 IP/设备/环境存在异常。当前 IP 为非常用 IP 或恶意 IP 段。 2011: 疑似非常用IP, 请求当前请求 IP 非该账号常用 IP。 2012: 疑似 IP 异常, 使用 IDC 机房 IP 或使用代理 IP 或使用恶意 IP 等。 205: 非公网有效 IP, 传进来的 IP 地址为内网 IP 地址或者 IP 保留地址。 示例值: [101,201]
ConstId	String	设备指纹ID, 如果集成了设备指纹, 并传入了正确的DeviceToken和Platform, 该字段正常输出; 如果DeviceToken异常(校验不通过), 则会在RiskType中返回"-1"标签, ConstId字段为空; 如果没有集成设备指纹ConstId字段默认为空。 示例值: 7BCF*****789
RiskInformation	String	风险扩展数据。 示例值: {"Model":[{"FieldName":"051***01","FieldValue":"2"}]}

## OutputModifyNameFront

修改黑白名单出参

被如下接口引用: ModifyNameList。

名称	类型	描述
Code	Integer	错误码, 0 表示成功, 非0表示失败错误码。0: 成功 1002: 参数错误 4300: 未开通服务 6000: 系统内部错误 示例值: 0
Message	String	错误信息 示例值: OK
Value	Array of String	空数组 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: []

## OutputModifyNameListFront

修改黑白名单数据出参

被如下接口引用: ModifyNameListData。

名称	类型	描述
Code	Integer	错误码, 0 表示成功, 非0表示失败错误码。0: 成功 1002: 参数错误 4300: 未开通服务 6000: 系统内部错误 示例值: 0
Message	String	错误信息 示例值: OK
Value	Array of String	空数组 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: []

## QQAccountInfo

QQ账号信息。

被如下接口引用：ManageMarketingRisk。

名称	类型	必选	描述
QQOpenId	String	是	QQ的OpenId。 示例值：A8E0232CD*****2058B0EA885
AppldUser	String	是	QQ分配给网站或应用的Appld，用来唯一标识网站或应用。 示例值：10*****60
AssociateAccount	String	否	用于标识QQ用户登录后所关联业务自身的账号ID。 示例值：100001
MobilePhone	String	否	账号绑定的MD5或SHA256加密的手机号。 注释：支持标准中国大陆11位手机号MD5加密后位的32位小写字母；支持标准中国大陆11位手机号SHA256加密后位的64位小写字母。 示例值：3ac9aa*****526ed9
DeviceId	String	否	用户设备号（已不推荐使用）。 示例值：095006*****dddc39

## SponsorInfo

网赚防刷相关参数

被如下接口引用：ManageMarketingRisk。

名称	类型	必选	描述
SponsorOpenId	String	否	助力场景建议填写：活动发起人微信OpenId。 示例值：ohmdTty_tw2*****igq9
SponsorDeviceNumber	String	否	助力场景建议填写：发起人设备号 示例值：095006*****dddc39
SponsorPhone	String	否	助力场景建议填写：发起人的MD5手机号 示例值：3ac9aa*****526ed9
SponsorIp	String	否	助力场景建议填写：发起人IP 示例值：125.*.*.*.235
CampaignUrl	String	否	助力场景建议填写：活动链接 示例值：https://*www.*.com/

## WeChatAccountInfo

微信账号信息。

被如下接口引用：ManageMarketingRisk。

名称	类型	必选	描述
WeChatOpenId	String	是	微信的OpenId/UnionId。 示例值：ohmdTty_tw2*****igq9
WeChatSubType	Integer	否	微信开放账号类型： 1：微信公众号/微信第三方登录。 2：微信小程序。 示例值：2
RandStr	String	否	随机串。如果WeChatSubType是2，该字段必填。Token签名随机数，建议16个字符。 示例值：1869ba9e74264061
WeChatAccessToken	String	否	如果WeChatSubType是1，填入授权的 access_token（注意：不是普通 access_token，详情请参阅官方说明文档。获取网页版本的 access_token 时，scope 字段必需填写snsapi_userinfo） 如果WeChatSubType是2，填入以session_key为密钥签名随机数RandStr（hmac_sha256签名算法）得到的字符串。 示例值：



名称	类型	必选	描述
			89_tP8ym_FpRC*****gWvQTBHBYDMY*****fP5pL2XhKq4t_xPd6Js2G3S-x9xTh1eIGRk73ipGaKfobaC2M
AssociateAccount	String	否	用于标识微信用户登录后所关联业务自身的账号ID。 示例值：100001
MobilePhone	String	否	账号绑定的MD5或SHA256加密的手机号。 注释：支持标准中国大陆11位手机号MD5加密后位的32位小写字符串； 支持标准中国大陆11位手机号SHA256加密后位的64位小写字符串。 示例值：3ac9aa*****526ed9
DeviceId	String	否	用户设备号（已不推荐使用）。 示例值：095006*****dddc39

# 错误码

最近更新時間：2024-12-20 01:48:42

## 功能說明

如果返回結果中存在 Error 字段，則表示調用 API 接口失敗。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 表示錯誤碼，Message 表示該錯誤的具體信息。

## 錯誤碼列表

### 公共錯誤碼

錯誤碼	說明
ActionOffline	接口已下線。
AuthFailure.InvalidAuthorization	請求頭部的 Authorization 不符合騰訊雲標準。
AuthFailure.InvalidSecretId	密鑰非法（不是雲 API 密鑰類型）。
AuthFailure.MFAFailure	MFA 錯誤。
AuthFailure.SecretIdNotFound	密鑰不存在。請在 <a href="#">控制台</a> 檢查密鑰是否已被刪除或者禁用，如狀態正常，請檢查密鑰是否填寫正確，注意前後不得有空格。
AuthFailure.SignatureExpire	簽名過期。Timestamp 和服務器時間相差不得超過五分鐘，請檢查本地時間是否和標準時間同步。
AuthFailure.SignatureFailure	簽名錯誤。簽名計算錯誤，請對照調用方式中的簽名方法文檔檢查簽名計算過程。
AuthFailure.TokenFailure	token 錯誤。
AuthFailure.UnauthorizedOperation	請求未授權。請參考 <a href="#">CAM</a> 文檔對鑒權的說明。
DryRunOperation	DryRun 操作，代表請求將會是成功的，只是多傳了 DryRun 參數。
FailedOperation	操作失敗。
InternalError	內部錯誤。
InvalidAction	接口不存在。
InvalidParameter	參數錯誤（包括參數格式、類型等錯誤）。
InvalidParameterValue	參數取值錯誤。
InvalidRequest	請求 body 的 multipart 格式錯誤。
IpInBlacklist	IP 地址在黑名單中。
IpNotInWhitelist	IP 地址不在白名單中。
LimitExceeded	超過配額限制。
MissingParameter	缺少參數。

错误码	说明
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。