全站加速网络

API 文档

腾讯云

文档目录

API 文档
更新历史

最近更新时间：2025-04-29 01:31:59

第 16 次发布

发布时间：2025-04-29 01:31:55

本次发布包含了以下内容：

改善已有的文档。

预下线接口：

- DescribeDomains
- DescribeDomainsConfig
- DescribeEcdnDomainLogs
- DescribeEcdnDomainStatistics
- DescribeEcdnStatistics
- DescribeIpStatus

第 15 次发布

发布时间：2025-03-28 01:32:15

本次发布包含了以下内容：

改善已有的文档。

预下线接口：

- DescribePurgeTasks
- PurgeUrlsCache

第 14 次发布

发布时间：2025-03-25 01:33:31

本次发布包含了以下内容：

改善已有的文档。

删除接口：

- AddEcdnDomain
- CreateVerifyRecord
- DeleteEcdnDomain
- DescribePurgeQuota
- PurgePathCache
- StartEcdnDomain
- StopEcdnDomain
- UpdateDomainConfig

删除数据结构：

- Quota

第 13 次发布

发布时间：2023-05-05 01:16:29

本次发布包含了以下内容：

改善已有的文档。

预下线接口：

- DescribePurgeQuota

第 12 次发布

发布时间：2023-04-27 01:19:24

本次发布包含了以下内容：

改善已有的文档。

预下线接口：

- AddEcdnDomain
- CreateVerifyRecord
- DeleteEcdnDomain
- PurgePathCache
- StartEcdnDomain
- StopEcdnDomain
- UpdateDomainConfig

第 11 次发布

发布时间：2022-01-18 08:09:09

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- DomainBriefInfo
  - 新增成员：Tag

第 10 次发布

发布时间：2021-12-22 08:11:17

本次发布包含了以下内容：

改善已有的文档。

新增数据结构：

- AdvanceHttps

修改数据结构：

- Origin
  - 新增成员：AdvanceHttps

第 9 次发布

发布时间：2021-05-28 08:01:48

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- CreateVerifyRecord

第 8 次发布

发布时间：2021-03-18 08:01:21

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- DescribeEcdnDomainStatistics
  - 新增入参：Area
- DescribeEcdnStatistics
  - 新增入参：Area

第 7 次发布

发布时间：2021-01-14 08:01:15

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- IpStatus
  - 新增成员：CreateTime

第 6 次发布

发布时间：2020-11-23 08:01:26

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- AddEcdnDomain
  - 新增入参：WebSocket
- UpdateDomainConfig
  - 新增入参：WebSocket

新增数据结构：

- WebSocket

修改数据结构：

- DomainDetailInfo
  - 新增成员：WebSocket

第 5 次发布

发布时间：2020-11-04 08:00:52

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- AddEcdnDomain
  - 新增入参：Tag

新增数据结构：

- Tag

修改数据结构：

- DomainDetailInfo
  - 新增成员：Tag

第 4 次发布

发布时间：2020-10-16 08:00:40

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DescribeIpStatus

新增数据结构：

- IpStatus

第 3 次发布

发布时间：2020-06-30 08:02:35

本次发布包含了以下内容：

改善已有的文档。

新增数据结构：

- Hsts

修改数据结构：

- Https
  - 新增成员：Hsts

第 2 次发布

发布时间：2020-04-16 08:01:06

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- Cache
  - 新增成员：FollowOrigin

第 1 次发布

发布时间：2020-03-17 15:54:37

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- AddEcdnDomain
- DeleteEcdnDomain
- DescribeDomains
- DescribeDomainsConfig
- DescribeEcdnDomainLogs
- DescribeEcdnDomainStatistics
- DescribeEcdnStatistics
- DescribePurgeQuota
- DescribePurgeTasks
- PurgePathCache
- PurgeUrlsCache
- StartEcdnDomain
- StopEcdnDomain
- UpdateDomainConfig

新增数据结构：

- Cache
- CacheKey
- CacheRule
- ClientCert
- DetailData
- DomainBriefInfo
- DomainData
- DomainDetailInfo
- DomainFilter
- DomainLogs
- EcdnData
- ForceRedirect
- HttpHeaderPathRule
- Https
- IpFilter
- IpFreqLimit
- Origin
- PurgeTask
- Quota
- ResourceData
- ResponseHeader
- ServerCert

- Sort
- TimestampData

简介

最近更新时间：2025-04-25 01:29:53

欢迎使用腾讯云全站加速网络（Enterprise Content Delivery Network）服务！

全站加速网络 ECDN 将静态边缘缓存与动态回源路径优化相融合，为纯动态网站与动静态混合型网站提供高可靠、低延时的一站式加速服务，解决了跨运营商、跨国、网络不稳定等因素导致的响应慢、丢包、服务不稳定等问题。

注意：

- 本章节全站加速网络 API 接口均为最新 API 3.0 接口，后续 ECDN 相关新增功能都会在此章节更新。我们强烈推荐您使用最新 API3.0 接口。
- 现有旧版 API 接口功能依然保持，未来可能停止维护，如您仍需使用旧版接口可参考：全站加速网络 API（旧版）简介。

API 概览

最近更新时间：2025-03-25 01:33:35

数据查询相关接口

接口名称	接口功能	频率限制（次/秒）
DescribeEcdnDomainStatistics	域名统计指标查询	20
DescribeEcdnStatistics	访问数据查询	20

服务查询相关接口

接口名称	接口功能	频率限制（次/秒）
DescribeIpStatus	查询平台服务节点IP	20

内容管理相关接口

接口名称	接口功能	频率限制（次/秒）
DescribePurgeTasks	刷新历史查询	20
PurgeUrlsCache	刷新 URL	20

日志查询相关接口

接口名称	接口功能	频率限制（次/秒）
DescribeEcdnDomainLogs	查询域名日志下载链接	20

配置管理相关接口

接口名称	接口功能	频率限制（次/秒）
DescribeDomains	查询域名基本信息	20
DescribeDomainsConfig	查询域名详细配置	20

⚠ 注意：

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

调用方式
请求结构

最近更新时间：2025-05-21 01:18:06

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

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

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

2. 通信协议

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

3. 请求方法

支持的 HTTP 请求方法：

- POST（推荐）
- GET

POST 请求支持的 Content-Type 类型：

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

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

4. 字符编码

均使用 `UTF-8` 编码。

公共参数

最近更新时间：2024-11-15 01:32:56

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

公共参数的具体内容会因您使用的签名方法版本不同而有所差异。

使用签名方法 v3 的公共参数

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

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

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

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

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

HTTP GET 请求结构示例：

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

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

HTTP POST（application/json）请求结构示例：

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

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

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

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

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

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

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

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

10
--58731222010402--
```

使用签名方法 v1 的公共参数

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

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

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

HTTP GET 请求结构示例：

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

Host: cvm.tencentcloudapi.com
```

HTTP POST 请求结构示例：

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

Host: cvm.tencentcloudapi.com
Content-Type: application/x-www-form-urlencoded

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

签名方法 v3

最近更新时间：2024-12-25 01:32:31

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区实例列表作为例子，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 中问号（?）后面的字符串内容，例如：Limit=10&Offset=0。 注意：CanonicalQueryString 需要参考 RFC3986 进行 URLEncode 编码（特殊字符编码后需大写字母），字符集 UTF-8。推荐使用编程语言标准库进行编码。
CanonicalHeaders	参与签名的头部信息，至少包含 host 和 content-type 两个头部，也可加入其他头部参与签名以提高自身请求的唯一性和安全性，此示例额外增加了接口名头部。 拼接规则： 1. 头部 key 和 value 统一转成小写，并去掉首尾空格，按照 key:value\n 格式拼接； 2. 多个头部，按照头部 key（小写）的 ASCII 升序进行拼接。

字段名称	解释
	此示例计算结果是 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。

字段名称	解释
HashedCanonicalRequest	前述步骤拼接所得规范请求串的哈希值，计算伪代码为 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 是二进制的数据，可能包含不可打印字符，将其转为十六进制字符串打印的输出分别为：da98fb70dcf6b112dc21038d1eeeb3a95c74b4dcb12c1131f864f6066bd02be0，8d70cbefb03939f929db64d32dc2ba89b1095620119fe3e050e2b18c5bd2752f，b596b923aad85185e2d1f6659d2a062e0a86731226e021e61bfe06f7ed05f5af。

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

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

2）计算签名，伪代码如下：

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

此示例计算结果是 10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f 。

4. 拼接 Authorization

按如下格式拼接 Authorization：

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

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

根据以上规则，示例中得到的值为：

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

最终完整的调用信息如下：

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

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

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

签名演示

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

- Python
- Java
- PHP
- Go
- NodeJS
- .NET
- C++
- Ruby

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

- Signature Demo

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

Java

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

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\": \"instan
ce-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" + hashedCanonicalReques
t;
System.out.println(stringToSign);

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

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

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

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

Python

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

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

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

# ************* 步骤 1：拼接规范请求串 *************
```

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

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


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

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

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

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

Golang

```
package main

import (
"crypto/hmac"
"crypto/sha256"
"encoding/hex"
"fmt"
"os"
"strings"
"time"
)

func sha256hex(s string) string {
b := sha256.Sum256([]byte(s))
return hex.EncodeToString(b[:])
}

func hmacsha256(s, key string) string {
hashed := hmac.New(sha256.New, []byte(key))
hashed.Write([]byte(s))
return string(hashed.Sum(nil))
}

func main() {
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secretId := os.Getenv("TENCENTCLOUD_SECRET_ID")
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
secretKey := os.Getenv("TENCENTCLOUD_SECRET_KEY")
host := "cvm.tencentcloudapi.com"
algorithm := "TC3-HMAC-SHA256"
service := "cvm"
version := "2017-03-12"
action := "DescribeInstances"
region := "ap-guangzhou"
//var timestamp int64 = time.Now().Unix()
var timestamp int64 = 1551113065

// step 1: build canonical request string
httpRequestMethod := "POST"
canonicalURI := "/"
canonicalQueryString := ""
canonicalHeaders := fmt.Sprintf("content-type:%s\nhost:%s\nx-tc-action:%s\n",
"application/json; charset=utf-8", host, strings.ToLower(action))
signedHeaders := "content-type;host;x-tc-action"
```

```
payload := `{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}`
hashedRequestPayload := sha256hex(payload)
canonicalRequest := fmt.Sprintf("%s\n%s\n%s\n%s\n%s\n%s",
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
<?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;
```

```php
// 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

```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.downc
ase}\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.AwayFromZer
o) / 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\": \"instanc
e-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\": \"instan
ce-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" + hashedCanonical
Request;
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++)
{
snprintf(buf, sizeof(buf), "%02x", hash[i]);
strcat(result, buf);
}
}

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

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

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


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

}

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

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

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

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

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

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

错误码	错误描述
AuthFailure.SecretIdNotFound	密钥不存在。请到控制台查看密钥是否被禁用，是否少复制了字符或者多了字符。
AuthFailure.SignatureFailure	签名错误。可能是签名计算错误，或者签名与实际发送的内容不相符合，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

最近更新时间：2024-12-25 01:32:31

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

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

推荐使用 API Explorer　点击调试

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

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

1. 申请安全凭证

在第一次使用云 API 之前，请前往 云 API 密钥页面 申请安全凭证。
安全凭证包括 SecretId 和 SecretKey：

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

申请安全凭证的具体步骤如下：

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

注意：每个账号最多可以拥有两对 SecretId/SecretKey。

2. 生成签名串

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

假设用户的 SecretId 和 SecretKey 分别是：

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

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

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

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

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

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

2.1. 对参数排序

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

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

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

2.2. 拼接请求字符串

此步骤生成请求字符串。
将把上一步排序好的请求参数格式化成“参数名称=参数值”的形式，如对 Action 参数，其参数名称为 "Action" ，参数值为 "DescribeInstances" ，因此格式化后就为 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&Secre
tId=AKID******************************&Timestamp=1465185768&Version=2017-03-12
```

2.3. 拼接签名原文字符串

此步骤生成签名原文字符串。
签名原文字符串由以下几个参数构成:

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

4. 请求字符串: 即上一步生成的请求字符串。

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

示例的拼接结果为：

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

2.4. 生成签名串

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

具体代码如下，以 PHP 语言为例：

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

最终得到的签名串为：

```
7RAM2xfNMO9EiVTNmPg06MRnCvQ=
```

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

3. 签名串编码

生成的签名串并不能直接作为请求参数，需要对其进行 URL 编码。

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

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

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

5. 签名演示

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

- Python
- Java
- PHP
- Go
- NodeJS
- .NET
- C++
- Ruby

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

- Signature Demo

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

最终输出的 url 可能为：`https://cvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKID********************************&Signature=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"), "HmacSHA
1")); // 公共参数
System.out.println(getUrl(params));
}
}
```

Python

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

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

import requests

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

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

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

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

Golang

```
package main

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

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

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

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

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

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

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

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

PHP

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

ksort($param);

$signStr = "GETcvm.tencentcloudapi.com/?";
foreach ( $param 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
// $param["Signature"] = $signature;
// $url = "https://cvm.tencentcloudapi.com/?".http_build_query($param);
// 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

```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

```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 requestMetho
d, 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.Ordin
al);
string sigInParam = MakeSignPlainText(headers, "GET", endpoint, "/");
string sigOutParam = Sign(SECRET_KEY, sigInParam);
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. 拼接签名原文字符串
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 01:28:52

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 06:29:48

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。

配置管理相关接口
查询域名详细配置

最近更新时间：2025-04-29 01:31:58

1. 接口描述

此接口处于预下线状态 。

预计下线时间：2025-07-27 10:25:12

接口请求域名： ecdn.tencentcloudapi.com 。

ECDN平台下线，接口开始预下线处理

本接口（DescribeDomainsConfig）用于查询CDN加速域名详细配置信息。

> 说明：
> 若您的业务已迁移至 CDN 控制台，请参考 CDN 接口文档，使用 CDN 相关API 进行操作。

默认接口请求频率限制：20次/秒。

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：DescribeDomainsConfig。
Version	是	String	公共参数，本接口取值：2019-10-12。
Region	否	String	公共参数，本接口不需要传递此参数。
Offset	否	Integer	分页查询的偏移地址，默认0。 示例值：0
Limit	否	Integer	分页查询的域名个数，默认100。 示例值：10
Filters.N	否	Array of DomainFilter	查询条件过滤器。
Sort	否	Sort	查询结果排序规则。

3. 输出参数

参数名称	类型	描述
Domains	Array of DomainDetailInfo	域名列表。
TotalCount	Integer	符合查询条件的域名总数，用于分页查询。 示例值：10
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 查询域名详细配置

输入示例

```
https://ecdn.tencentcloudapi.com/?Action=DescribeDomainsConfig
&<公共请求参数>
```

输出示例


```
{
"Response": {
"RequestId": "932fe708-0ce1-46ec-b403-bcd8bdb08fdd",
"Domains": [
{
"AppId": 1251000000,
"Area": "mainland",
"Cache": {
"CacheRules": [
{
"CacheType": "all",
"CacheContents": [
"*"
],
"CacheTime": 0
},
{
"CacheType": "file",
"CacheContents": [
"gif",
"png",
"bmp",
"jpg",
"jpeg",
"mp3",
"wma",
"flv",
"mp4",
"wmv",
"avi",
"m3u8",
"ts"
],
"CacheTime": 86400
},
{
"CacheType": "file",
"CacheContents": [
"doc",
```

```
"docx",
"xls",
"xlsx",
"ppt",
"pptx",
"txt",
"pdf"
],
"CacheTime": 86400
},
{
"CacheType": "file",
"CacheContents": [
"exe",
"apk",
"ipa",
"rar",
"zip",
"7z",
"css",
"js",
"xml",
"ini",
"swf",
"ico"
],
"CacheTime": 86400
}
]
},
"CacheKey": {
"FullUrlCache": "on"
},
"Cname": "test.com.com.dsa.dnsv1.com",
"CreateTime": "2019-12-03 15:23:50",
"Disable": "normal",
"Domain": "test.com",
"ForceRedirect": null,
"Https": {
"Switch": "off",
"Http2": "off",
"Spdy": "off",
"OcspStapling": "off",
"VerifyClient": "off",
"CertInfo": null,
"ClientCertInfo": null,
"SslStatus": "closed"
},
"IpFilter": {
"Switch": "off",
```

```
"FilterType": "blacklist",
"Filters": []
},
"IpFreqLimit": {
"Switch": "off",
"Qps": null
},
"Origin": {
"Origins": [
"1.1.1.1"
],
"OriginType": "ip",
"ServerName": null,
"OriginPullProtocol": "http",
"BackupOrigins": [],
"BackupOriginType": null
},
"ProjectId": 0,
"Readonly": "normal",
"ResourceId": "ecdn-xxxxxx",
"ResponseHeader": {
"HeaderRules": [],
"Switch": "off"
},
"Status": "processing",
"UpdateTime": "2019-12-03 15:23:50",
"Tag": [],
"WebSocket": null
}
],
"TotalCount": 10
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee

- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.EcdnConfigError	域名配置更新操作失败，请重试或联系客服人员解决。
InternalError.CamSystemError	CAM鉴权错误，请稍后重试。
InternalError.EcdnConfigError	内部配置服务错误，请重试或联系客服人员解决。
InternalError.EcdnDbError	内部数据错误，请联系腾讯云工程师进一步排查。
InternalError.EcdnQuerySystemError	系统内部错误，请联系客户人员解决或稍后重试。
InternalError.EcdnSystemError	系统错误，请联系腾讯云工程师进一步排查。
InternalError.Error	内部服务错误，请联系腾讯云工程师进一步排查。
InvalidParameter.EcdnCertNoCertInfo	无法解析证书信息。
InvalidParameter.EcdnConfigInvalidCache	缓存配置不合法 。
InvalidParameter.EcdnInterfaceError	内部接口错误，请联系腾讯云工程师进一步排查。
InvalidParameter.EcdnParamError	参数错误，请参考文档中示例参数填充。
LimitExceeded.EcdnDomainOpTooOften	域名操作过于频繁。
ResourceNotFound.EcdnDomainNotExists	账号下无此域名，请确认后重试。
ResourceNotFound.EcdnUserNotExists	未开通ECDN服务，请开通后使用此接口。
UnauthorizedOperation.EcdnCamUnauthorized	子账号未配置cam策略。

查询域名基本信息

最近更新时间：2025-04-29 01:31:58

1. 接口描述

此接口处于预下线状态 。

预计下线时间：2025-07-27 10:25:12

接口请求域名： ecdn.tencentcloudapi.com 。

ECDN平台下线，接口开始预下线处理

本接口（DescribeDomains）用于查询CDN域名基本信息，包括项目id，状态，业务类型，创建时间，更新时间等。

> 说明：
> 若您的业务已迁移至 CDN 控制台，请参考 CDN 接口文档，使用 CDN 相关API 进行操作。

默认接口请求频率限制：20次/秒。

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：DescribeDomains。
Version	是	String	公共参数，本接口取值：2019-10-12。
Region	否	String	公共参数，本接口不需要传递此参数。
Offset	否	Integer	分页查询的偏移地址，默认0。 示例值：0
Limit	否	Integer	分页查询的域名个数，默认100，最大支持1000。 示例值：10
Filters.N	否	Array of DomainFilter	查询条件过滤器。

3. 输出参数

参数名称	类型	描述
Domains	Array of DomainBriefInfo	域名信息列表。
TotalCount	Integer	域名总个数。 示例值：10
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 查询域名简要信息

输入示例

```
https://ecdn.tencentcloudapi.com/?Action=DescribeDomains
&<公共请求参数>
```

输出示例


```
{
"Response": {
"RequestId": "104fcdb5-293c-4f6f-b63d-0c9e430264e3",
"Domains": [
{
"AppId": 1251000000,
"Area": "mainland",
"Cname": "test.com.dsa.dnsv1.com",
"CreateTime": "2019-12-03 15:23:50",
"Disable": "normal",
"Domain": "test.com",
"Tag": [],
"Origin": {
"Origins": [
"1.1.1.1"
],
"OriginType": "ip",
"ServerName": null,
"OriginPullProtocol": "http",
"BackupOrigins": [],
"BackupOriginType": null
},
"ProjectId": 0,
"Readonly": "normal",
"ResourceId": "ecdn-xxxx",
"Status": "processing",
"UpdateTime": "2019-12-03 15:23:50"
}
],
"TotalCount": 10
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError.CamSystemError	CAM鉴权错误，请稍后重试。
InternalError.EcdnConfigError	内部配置服务错误，请重试或联系客服人员解决。
InternalError.EcdnDbError	内部数据错误，请联系腾讯云工程师进一步排查。
InternalError.EcdnQuerySystemError	系统内部错误，请联系客户人员解决或稍后重试。
InternalError.EcdnSystemError	系统错误，请联系腾讯云工程师进一步排查。
InternalError.Error	内部服务错误，请联系腾讯云工程师进一步排查。
InternalError.ProxyServer	后端服务错误,请稍后重试 。
InvalidParameter.EcdnInterfaceError	内部接口错误，请联系腾讯云工程师进一步排查。
InvalidParameter.EcdnParamError	参数错误，请参考文档中示例参数填充。
LimitExceeded.EcdnDomainOpTooOften	域名操作过于频繁。
ResourceNotFound.EcdnDomainNotExists	账号下无此域名，请确认后重试。
ResourceNotFound.EcdnUserNotExists	未开通ECDN服务，请开通后使用此接口。
UnauthorizedOperation.EcdnCamUnauthorized	子账号未配置cam策略。
UnauthorizedOperation.OperationTooOften	操作过于频繁，请稍后重试 。

日志查询相关接口
查询域名日志下载链接

最近更新时间：2025-04-29 01:31:58

1. 接口描述

此接口处于预下线状态 。

预计下线时间：2025-07-27 10:25:12

接口请求域名： ecdn.tencentcloudapi.com 。

ECDN平台下线，接口开始预下线处理

本接口（DescribeEcdnDomainLogs）用于查询域名的访问日志下载地址。

默认接口请求频率限制：20次/秒。

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：DescribeEcdnDomainLogs。
Version	是	String	公共参数，本接口取值：2019-10-12。
Region	否	String	公共参数，本接口不需要传递此参数。
Domain	是	String	待查询域名。 示例值：www.test.com
StartTime	是	Timestamp	日志起始时间。如：2019-10-01 00:00:00 示例值：2019-09-04 00:00:00
EndTime	是	Timestamp	日志结束时间，只支持最近30天内日志查询。2019-10-02 00:00:00 示例值：2019-09-04 12:00:00
Offset	否	Integer	日志链接列表分页起始地址，默认0。 示例值：0
Limit	否	Integer	日志链接列表分页记录条数，默认100，最大1000。 示例值：1

3. 输出参数

参数名称	类型	描述
DomainLogs	Array of DomainLogs	日志链接列表。 注意：此字段可能返回 null，表示取不到有效值。
TotalCount	Integer	日志链接总条数。 示例值：300
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 查询域名日志下载链接

输入示例

```
https://ecdn.tencentcloudapi.com/?Action=DescribeEcdnDomainLogs
&StartTime=2019-09-04 00:00:00
&EndTime=2019-09-04 12:00:00
&Domain=www.test.com
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "13d41d37-546f-42ed-a3b9-ff82a51ecd0a",
"DomainLogs": [
{
"StartTime": "2019-09-04 23:00:00",
"EndTime": "2019-09-04 23:59:59",
"LogPath": "http://www.test.qcloud.com/20190904/23/201909042300-www.test.com.gz?st=hGzJr0QFpo3jYM2uj7kkjA&e
=3135214538"
}
],
"TotalCount": 300
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError.EcdnConfigError	内部配置服务错误，请重试或联系客服人员解决。
InternalError.EcdnDbError	内部数据错误，请联系腾讯云工程师进一步排查。
InvalidParameter.EcdnParamError	参数错误，请参考文档中示例参数填充。
InvalidParameter.EcdnStatInvalidDate	日期不合法，请参考文档中日期示例。
LimitExceeded.EcdnDomainOpTooOften	域名操作过于频繁。
ResourceNotFound.EcdnDomainNotExists	账号下无此域名，请确认后重试。
ResourceNotFound.EcdnHostNotExists	账号下无此域名，请确认后重试。
ResourceNotFound.EcdnUserNotExists	未开通ECDN服务，请开通后使用此接口。
UnauthorizedOperation.EcdnAccountUnauthorized	子账号禁止查询整体数据。
UnauthorizedOperation.EcdnCamUnauthorized	子账号未配置cam策略。
UnauthorizedOperation.EcdnDomainUnauthorized	ECDN子账号加速域名未授权。
UnauthorizedOperation.EcdnHostUnauthorized	ECDN子账号加速域名未授权。
UnauthorizedOperation.EcdnNoDomainUnauthorized	子账号没有授权域名权限，请授权后重试。

数据查询相关接口
访问数据查询

最近更新时间：2025-04-29 01:31:59

1. 接口描述

此接口处于预下线状态 。

预计下线时间：2025-07-27 10:25:12

接口请求域名： ecdn.tencentcloudapi.com 。

ECDN平台下线，接口开始预下线处理

DescribeEcdnStatistics用于查询 ECDN 实时访问监控数据，支持以下指标查询：

- 流量（单位为 byte）
- 带宽（单位为 bps）
- 请求数（单位为 次）
- 状态码 2xx 汇总及各 2 开头状态码明细（单位为 个）
- 状态码 3xx 汇总及各 3 开头状态码明细（单位为 个）
- 状态码 4xx 汇总及各 4 开头状态码明细（单位为 个）
- 状态码 5xx 汇总及各 5 开头状态码明细（单位为 个）

默认接口请求频率限制：20次/秒。

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：DescribeEcdnStatistics。
Version	是	String	公共参数，本接口取值：2019-10-12。
Region	否	String	公共参数，本接口不需要传递此参数。
StartTime	是	Timestamp	查询起始时间，如：2019-12-13 00:00:00 示例值：2018-09-04 00:00:00
EndTime	是	Timestamp	查询结束时间，如：2019-12-13 23:59:59 示例值：2018-09-04 12:00:00
Metrics.N	是	Array of String	指定查询指标，支持的类型有： flux：流量，单位为 byte bandwidth：带宽，单位为 bps request：请求数，单位为 次 2xx：返回 2xx 状态码汇总或者 2 开头状态码数据，单位为 个 3xx：返回 3xx 状态码汇总或者 3 开头状态码数据，单位为 个 4xx：返回 4xx 状态码汇总或者 4 开头状态码数据，单位为 个 5xx：返回 5xx 状态码汇总或者 5 开头状态码数据，单位为 个 示例值：["flux\n"]
Interval	是	Integer	时间粒度，支持以下几种模式： 1 天 1，5，15，30，60，120，240，1440 2 ~ 3 天 15，30，60，120，240，1440 4 ~ 7 天 30，60，120，240，1440

参数名称	必选	类型	描述
			8 ~ 31 天 60，120，240，1440 示例值：60
Domains.N	否	Array of String	指定查询域名列表 最多可一次性查询30个加速域名。 示例值：["www.test.com\n"]
Projects.N	否	Array of Integer	指定要查询的项目 ID，前往查看项目 ID 未填充域名情况下，指定项目查询，若填充了具体域名信息，以域名为主 示例值：["0\n"]
Area	否	String	统计区域: mainland: 境内 oversea: 境外 global: 全部 默认 global 示例值：global

3. 输出参数

参数名称	类型	描述
Data	Array of ResourceData	指定条件查询得到的数据明细
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 访问数据查询

输入示例

```
https://ecdn.tencentcloudapi.com/?Action=DescribeEcdnStatistics
&StartTime=2018-09-04 00:00:00
&EndTime=2018-09-04 12:00:00
&Metrics.0=flux
&Interval=60
&Domains.0=www.test.com
&Projects.0=0
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "13d41d37-546f-42ed-a3b9-ff82a51ecd0a",
"Data": [
{
"Resource": "all",
```

```
"EcdnData": {
"Metrics": [
"flux",
"request"
],
"DetailData": [
{
"Time": "2019-12-13 00:00:00",
"Value": [
10,
20
]
},
{
"Time": "2019-12-13 00:05:00",
"Value": [
20,
30
]
}
]
}
}
]
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError.EcdnConfigError	内部配置服务错误，请重试或联系客服人员解决。
InternalError.EcdnDbError	内部数据错误，请联系腾讯云工程师进一步排查。
InternalError.EcdnSystemError	系统错误，请联系腾讯云工程师进一步排查。
InternalError.ProxyServer	后端服务错误,请稍后重试 。
InvalidParameter.EcdnInvalidParamInterval	统计粒度不合法，请参考文档中统计分析示例。
InvalidParameter.EcdnParamError	参数错误，请参考文档中示例参数填充。
InvalidParameter.EcdnStatInvalidDate	日期不合法，请参考文档中日期示例。
InvalidParameter.EcdnStatInvalidMetric	统计类型不合法，请参考文档中统计分析示例。
InvalidParameter.ParamError	参数错误。
LimitExceeded.EcdnDomainOpTooOften	域名操作过于频繁。
ResourceNotFound.EcdnDomainNotExists	账号下无此域名，请确认后重试。
ResourceNotFound.EcdnHostNotExists	账号下无此域名，请确认后重试。
ResourceNotFound.EcdnProjectNotExists	项目不存在。
ResourceNotFound.EcdnUserNotExists	未开通ECDN服务，请开通后使用此接口。
UnauthorizedOperation.CdnAccountUnauthorized	子账号禁止查询整体数据。
UnauthorizedOperation.CdnCamUnauthorized	子账号未配置cam策略。
UnauthorizedOperation.CdnDomainUnauthorized	ECDN子账号加速域名未授权。
UnauthorizedOperation.CdnHostUnauthorized	ECDN子账号加速域名未授权。
UnauthorizedOperation.CdnNoDomainUnauthorized	子账号没有授权域名权限，请授权后重试。
UnauthorizedOperation.CdnProjectUnauthorized	子账号项目未授权。
UnauthorizedOperation.DomainNoPermission	ECDN 子账号加速域名未授权。
UnauthorizedOperation.DomainsNoPermission	ECDN 子账号加速域名未授权。
UnauthorizedOperation.EcdnAccountUnauthorized	子账号禁止查询整体数据。
UnauthorizedOperation.EcdnCamUnauthorized	子账号未配置cam策略。
UnauthorizedOperation.EcdnDomainUnauthorized	ECDN子账号加速域名未授权。
UnauthorizedOperation.EcdnHostUnauthorized	ECDN子账号加速域名未授权。
UnauthorizedOperation.EcdnNoDomainUnauthorized	子账号没有授权域名权限，请授权后重试。
UnauthorizedOperation.EcdnProjectUnauthorized	子账号项目未授权。
UnauthorizedOperation.NoPermission	ECDN 子账号cam未授权。

错误码	描述
UnauthorizedOperation.ProjectNoPermission	ECDN 子账号项目未授权。
UnauthorizedOperation.ProjectsNoPermission	ECDN 子账号项目未授权。
UnauthorizedOperation.Unknown	未知错误,请稍后重试 。

域名统计指标查询

最近更新时间：2025-04-29 01:31:59

1. 接口描述

此接口处于预下线状态 。

预计下线时间：2025-07-27 10:25:12

接口请求域名： ecdn.tencentcloudapi.com 。

ECDN平台下线，接口开始预下线处理

本接口（DescribeEcdnDomainStatistics）用于查询指定时间段内的域名访问统计指标。

> 说明：
> 若您的业务已迁移至 CDN 控制台，请参考 CDN 接口文档，使用 CDN 相关API 进行操作。

默认接口请求频率限制：20次/秒。

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：DescribeEcdnDomainStatistics。
Version	是	String	公共参数，本接口取值：2019-10-12。
Region	否	String	公共参数，本接口不需要传递此参数。
StartTime	是	Timestamp	查询起始时间，如：2019-12-13 00:00:00。 起止时间不超过90天。 示例值：2018-09-04 00:00:00
EndTime	是	Timestamp	查询结束时间，如：2019-12-13 23:59:59。 起止时间不超过90天。 示例值：2018-09-04 12:00:00
Metrics.N	是	Array of String	统计指标名称: flux：流量，单位为 byte bandwidth：带宽，单位为 bps request：请求数，单位为 次 示例值：["flux\n","delay\n","request\n","bandwidth\n"]
Domains.N	否	Array of String	指定查询域名列表 示例值：["www.test.com\n"]
Projects.N	否	Array of Integer	指定要查询的项目 ID，前往查看项目 ID 未填充域名情况下，指定项目查询，若填充了具体域名信息，以域名为主 示例值：["0\n"]
Offset	否	Integer	列表分页起始地址，默认0。 示例值：0
Limit	否	Integer	列表分页记录条数，默认1000，最大3000。 示例值：1000

参数名称	必选	类型	描述
Area	否	String	统计区域: mainland: 境内 oversea: 境外 global: 全部 默认 global 示例值：global

3. 输出参数

参数名称	类型	描述
Data	Array of DomainData	域名数据
TotalCount	Integer	数量 示例值：20
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 域名统计指标查询

输入示例

```
https://ecdn.tencentcloudapi.com/?Action=DescribeEcdnDomainStatistics
&StartTime=2018-09-04 00:00:00
&EndTime=2018-09-04 12:00:00
&Metrics.0=flux
&Metrics.1=delay
&Metrics.2=request
&Metrics.3=bandwidth
&Domains.0=www.test.com
&Projects.0=0
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "13d41d37-546f-42ed-a3b9-ff82a51ecd0a",
"Data": [
{
"Resource": "www.test.com",
"DetailData": [
{
"Name": "request",
"Value": 5628872958
},
```

```
{
"Name": "flux",
"Value": 3535122082980
},
{
"Name": "delay",
"Value": 87
},
{
"Name": "bandwidth",
"Value": 825782981
}
]
}
],
"TotalCount": 20
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError.EcdnDbError	内部数据错误，请联系腾讯云工程师进一步排查。
InternalError.EcdnSystemError	系统错误，请联系腾讯云工程师进一步排查。
InvalidParameter.EcdnParamError	参数错误，请参考文档中示例参数填充。
InvalidParameter.EcdnStatInvalidDate	日期不合法，请参考文档中日期示例。
InvalidParameter.EcdnStatInvalidMetric	统计类型不合法，请参考文档中统计分析示例。
InvalidParameter.ParamError	参数错误。
LimitExceeded.EcdnDomainOpTooOften	域名操作过于频繁。
ResourceNotFound.EcdnDomainNotExists	账号下无此域名，请确认后重试。
ResourceNotFound.EcdnProjectNotExists	项目不存在。
ResourceNotFound.EcdnUserNotExists	未开通ECDN服务，请开通后使用此接口。
UnauthorizedOperation.CdnAccountUnauthorized	子账号禁止查询整体数据。
UnauthorizedOperation.CdnCamUnauthorized	子账号未配置cam策略。
UnauthorizedOperation.CdnDomainUnauthorized	ECDN子账号加速域名未授权。
UnauthorizedOperation.CdnHostUnauthorized	ECDN子账号加速域名未授权。
UnauthorizedOperation.CdnNoDomainUnauthorized	子账号没有授权域名权限，请授权后重试。
UnauthorizedOperation.CdnProjectUnauthorized	子账号项目未授权。
UnauthorizedOperation.DomainNoPermission	ECDN 子账号加速域名未授权。
UnauthorizedOperation.DomainsNoPermission	ECDN 子账号加速域名未授权。
UnauthorizedOperation.EcdnAccountUnauthorized	子账号禁止查询整体数据。
UnauthorizedOperation.EcdnCamUnauthorized	子账号未配置cam策略。
UnauthorizedOperation.EcdnDomainUnauthorized	ECDN子账号加速域名未授权。
UnauthorizedOperation.EcdnHostUnauthorized	ECDN子账号加速域名未授权。
UnauthorizedOperation.EcdnNoDomainUnauthorized	子账号没有授权域名权限，请授权后重试。
UnauthorizedOperation.EcdnProjectUnauthorized	子账号项目未授权。
UnauthorizedOperation.NoPermission	ECDN 子账号cam未授权。
UnauthorizedOperation.ProjectNoPermission	ECDN 子账号项目未授权。
UnauthorizedOperation.ProjectsNoPermission	ECDN 子账号项目未授权。

内容管理相关接口
刷新 URL

最近更新时间：2025-04-25 01:29:53

1. 接口描述

此接口处于预下线状态 。

预计下线时间：2025-06-25 11:44:09

接口请求域名： ecdn.tencentcloudapi.com 。

ECDN即将下线，如需要动态加速请使用EdgeOne

PurgeUrlsCache 用于批量刷新Url，一次提交将返回一个刷新任务id。

> 说明：
> 若您的业务已迁移至 CDN 控制台，请参考 CDN 接口文档，使用 CDN 相关API 进行操作。

默认接口请求频率限制：20次/秒。

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：PurgeUrlsCache。
Version	是	String	公共参数，本接口取值：2019-10-12。
Region	否	String	公共参数，本接口不需要传递此参数。
Urls.N	是	Array of String	要刷新的Url列表，必须包含协议头部。 示例值：["http://www.test.com\/1.jpg"]

3. 输出参数

参数名称	类型	描述
TaskId	String	刷新任务Id。 示例值：1533045796-i60rfmzm
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 刷新Url

输入示例

```
https://ecdn.tencentcloudapi.com/?Action=PurgeUrlsCache
&Urls.0=http://www.test.com/1.jpg
```

```
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "4d5a83f8-a61f-445b-8036-5636be640bef",
"TaskId": "1533045796-i60rfmzm"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.EcdnConfigError	域名配置更新操作失败，请重试或联系客服人员解决。
InternalError.EcdnConfigError	内部配置服务错误，请重试或联系客服人员解决。
InternalError.EcdnDbError	内部数据错误，请联系腾讯云工程师进一步排查。
InternalError.EcdnSystemError	系统错误，请联系腾讯云工程师进一步排查。
InvalidParameter.EcdnDomainInvalidStatus	域名状态不合法。

错误码	描述
InvalidParameter.EcdnInterfaceError	内部接口错误，请联系腾讯云工程师进一步排查。
InvalidParameter.EcdnParamError	参数错误，请参考文档中示例参数填充。
InvalidParameter.EcdnPurgeWildcardNotAllowed	刷新不支持泛域名。
InvalidParameter.EcdnUrlExceedLength	URL 超过限制长度。
LimitExceeded.EcdnPurgeUrlExceedBatchLimit	刷新的Url数量超过单次限制。
LimitExceeded.EcdnPurgeUrlExceedDayLimit	刷新的Url数量超过每日限额。
ResourceNotFound.EcdnDomainNotExists	账号下无此域名，请确认后重试。
ResourceNotFound.EcdnUserNotExists	未开通ECDN服务，请开通后使用此接口。
UnauthorizedOperation.EcdnCamUnauthorized	子账号未配置cam策略。
UnauthorizedOperation.EcdnDomainUnauthorized	ECDN子账号加速域名未授权。
UnauthorizedOperation.EcdnMigratedCdn	请前往CDN控制台进行操作 。
UnauthorizedOperation.EcdnUserIsSuspended	加速服务已停服，请重启加速服务后重试。

刷新历史查询

最近更新时间：2025-04-25 01:29:53

1. 接口描述

此接口处于预下线状态 。

预计下线时间：2025-06-25 11:44:09

接口请求域名： ecdn.tencentcloudapi.com 。

ECDN即将下线，如需要动态加速请使用EdgeOne

DescribePurgeTasks 用于查询刷新任务提交历史记录及执行进度。

> 说明：
> 若您的业务已迁移至 CDN 控制台，请参考 CDN 接口文档，使用 CDN 相关API 进行操作。

默认接口请求频率限制：20次/秒。

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：DescribePurgeTasks。
Version	是	String	公共参数，本接口取值：2019-10-12。
Region	否	String	公共参数，本接口不需要传递此参数。
PurgeType	否	String	查询刷新类型。url：查询 url 刷新记录；path：查询目录刷新记录。 示例值：ur
StartTime	否	Timestamp	开始时间，如2018-08-08 00:00:00。
EndTime	否	Timestamp	结束时间，如2018-08-08 23:59:59。
TaskId	否	String	提交时返回的任务 Id，查询时 TaskId 和起始时间必须指定一项。 示例值：1234567
Offset	否	Integer	分页查询偏移量，默认为0（从第0条开始）。 示例值：0
Limit	否	Integer	分页查询限制数目，默认为20。 示例值：10
Keyword	否	String	查询关键字，请输入域名或 http(s):// 开头完整 URL。
Status	否	String	查询指定任务状态，fail表示失败，done表示成功，process表示刷新中。

3. 输出参数

参数名称	类型	描述
PurgeLogs	Array of PurgeTask	刷新历史记录。

参数名称	类型	描述
TotalCount	Integer	任务总数，用于分页。 示例值：20
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 刷新历史查询

输入示例

```
https://ecdn.tencentcloudapi.com/?Action=DescribePurgeTasks
&PurgeType=ur
&TaskId=1234567
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "4d5a83f8-a61f-445b-8036-5636be640bef",
"PurgeLogs": [
{
"TaskId": "153303185323131331",
"Url": "http://www.test.com/",
"Status": "Done",
"PurgeType": "url",
"FlushType": "flush",
"CreateTime": "2018-07-30 18:10:53"
}
],
"TotalCount": 20
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError.EcdnSystemError	系统错误，请联系腾讯云工程师进一步排查。
InvalidParameter.EcdnInterfaceError	内部接口错误，请联系腾讯云工程师进一步排查。
InvalidParameter.EcdnParamError	参数错误，请参考文档中示例参数填充。
ResourceNotFound.EcdnDomainNotExists	账号下无此域名，请确认后重试。
ResourceNotFound.EcdnUserNotExists	未开通ECDN服务，请开通后使用此接口。
UnauthorizedOperation.EcdnCamUnauthorized	子账号未配置cam策略。
UnauthorizedOperation.EcdnMigratedCdn	请前往CDN控制台进行操作 。
UnauthorizedOperation.EcdnUserIsSuspended	加速服务已停服，请重启加速服务后重试。

服务查询相关接口
查询平台服务节点IP

最近更新时间：2025-04-29 01:31:58

1. 接口描述

此接口处于预下线状态 。

预计下线时间：2025-07-27 10:25:12

接口请求域名： ecdn.tencentcloudapi.com 。

ECDN平台下线，接口开始预下线处理

DescribeIpStatus 用于查询域名所在加速平台的所有节点信息, 如果您的源站有白名单设置,可以通过本接口获取ECDN服务的节点IP进行加白, 本接口为内测接口,请联系腾讯云工程师开白。

由于产品服务节点常有更新，对于源站开白的使用场景，请定期调用接口获取最新节点信息，若新增服务节点发布7日后您尚未更新加白导致回源失败等问题，ECDN侧不对此承担责任。

默认接口请求频率限制：20次/秒。

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：DescribeIpStatus。
Version	是	String	公共参数，本接口取值：2019-10-12。
Region	否	String	公共参数，本接口不需要传递此参数。
Domain	是	String	加速域名 示例值：www.test.com
Area	否	String	查询区域： mainland: 国内节点 overseas: 海外节点 global: 全球节点 示例值：global

3. 输出参数

参数名称	类型	描述
Ips	Array of IpStatus	节点列表
TotalCount	Integer	节点总个数 示例值：0
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 查询域名节点信息

输入示例

```
https://ecdn.tencentcloudapi.com/?Action=DescribeIpStatus
&Domain=www.test.com
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "b6e9964d-26a3-49d0-adab-993e17d2f950",
"Ips": [
{
"Ip": "1.1.1.1",
"District": "广东",
"Isp": "电信",
"City": "深圳",
"Status": "online",
"CreateTime": "2019-10-12 00:00:00"
}
],
"TotalCount": 0
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError.EcdnConfigError	内部配置服务错误，请重试或联系客服人员解决。
InternalError.EcdnDbError	内部数据错误，请联系腾讯云工程师进一步排查。
InternalError.EcdnSystemError	系统错误，请联系腾讯云工程师进一步排查。
InvalidParameter.EcdnParamError	参数错误，请参考文档中示例参数填充。
ResourceNotFound.EcdnDomainNotExists	账号下无此域名，请确认后重试。
ResourceNotFound.EcdnHostNotExists	账号下无此域名，请确认后重试。
ResourceNotFound.EcdnUserNotExists	未开通ECDN服务，请开通后使用此接口。
UnauthorizedOperation.EcdnCamUnauthorized	子账号未配置cam策略。
UnauthorizedOperation.EcdnDomainUnauthorized	ECDN子账号加速域名未授权。
UnauthorizedOperation.EcdnHostUnauthorized	ECDN子账号加速域名未授权。
UnauthorizedOperation.EcdnUserNoWhitelist	非内测白名单用户，无该功能使用权限。
UnauthorizedOperation.OperationTooOften	操作过于频繁，请稍后重试 。

数据结构

最近更新时间：2025-03-25 01:33:34

AdvanceHttps

回源的自定义Https配置

被如下接口引用：DescribeDomains, DescribeDomainsConfig。

名称	类型	必选	描述
CustomTlsStatus	String	否	自定义Tls数据开关 注意：此字段可能返回 null，表示取不到有效值。 示例值：on
TlsVersion	Array of String	否	Tls版本列表，支持设置 TLSv1, TLSV1.1, TLSV1.2, TLSv1.3，修改时必须开启连续的版本 注意：此字段可能返回 null，表示取不到有效值。 示例值：["TLSv1","TLSv1.1","TLSv1.2","TLSv1.3"]
Cipher	String	否	自定义加密套件 注意：此字段可能返回 null，表示取不到有效值。 示例值："EECDH+AES128"
VerifyOriginType	String	否	回源双向校验开启状态 off - 关闭校验 oneWay - 校验源站 twoWay - 双向校验 注意：此字段可能返回 null，表示取不到有效值。 示例值：off
CertInfo	ServerCert	否	回源层证书配置信息 注意：此字段可能返回 null，表示取不到有效值。
OriginCertInfo	ClientCert	否	源站证书配置信息 注意：此字段可能返回 null，表示取不到有效值。

Cache

缓存配置简单版本，该版本不支持设置源站未返回max-age情况下的缓存规则。

被如下接口引用：DescribeDomainsConfig。

名称	类型	必选	描述
CacheRules	Array of CacheRule	是	缓存配置规则数组。
FollowOrigin	String	否	遵循源站 Cache-Control: max-age 配置，白名单功能。 on：开启 off：关闭 开启后，未能匹配 CacheRules 规则的资源将根据源站返回的 max-age 值进行节点缓存；匹配了 CacheRules 规则的资源将按照 CacheRules 中设置的缓存过期时间在节点进行缓存 注意：此字段可能返回 null，表示取不到有效值。

CacheKey

缓存相关配置。

被如下接口引用：DescribeDomainsConfig。

名称	类型	必选	描述
FullUrlCache	String	否	是否开启全路径缓存，on或off。

CacheRule

缓存配置规则。

被如下接口引用：DescribeDomainsConfig。

名称	类型	必选	描述
CacheType	String	是	缓存类型，支持all，file，directory，path，index，分别表示全部文件，后缀类型，目录，完整路径，首页。
CacheContents	Array of String	是	缓存内容列表。
CacheTime	Integer	是	缓存时间，单位秒。

ClientCert

https客户端证书配置。

被如下接口引用：DescribeDomains, DescribeDomainsConfig。

名称	类型	必选	描述
Certificate	String	是	客户端证书，pem格式。 注意：此字段可能返回 null，表示取不到有效值。
CertName	String	否	客户端证书名称。 注意：此字段可能返回 null，表示取不到有效值。
ExpireTime	Timestamp	否	证书过期时间。 注意：此字段可能返回 null，表示取不到有效值。
DeployTime	Timestamp	否	证书颁发时间。 注意：此字段可能返回 null，表示取不到有效值。

DetailData

排序类型的数据结构

被如下接口引用：DescribeEcdnDomainStatistics。

名称	类型	描述
Name	String	数据类型的名称
Value	Float	数据值

DomainBriefInfo

CDN域名简要信息。

被如下接口引用：DescribeDomains。

名称	类型	描述
ResourceId	String	域名ID。 示例值：ecdn-xxxx
AppId	Integer	腾讯云账号ID。 示例值：1251000000
Domain	String	CDN加速域名。 示例值：test.com
Cname	String	域名CName。 示例值：test.com.dsa.dnsv1.com
Status	String	域名状态，pending，rejected，processing， online，offline，deleted分别表示审核中，审核未通过，审核通过部署中，已开启，已关闭，已删除。 示例值：processing
ProjectId	Integer	项目ID。 示例值：0
CreateTime	Timestamp	域名创建时间。 示例值：2019-12-03 15:23:50
UpdateTime	Timestamp	域名更新时间。 示例值：2019-12-03 15:23:50
Origin	Origin	源站配置详情。 示例值：{"Origins":["1.1.1.1"]}
Disable	String	域名封禁状态，normal，overdue，quota，malicious，ddos，idle，unlicensed，capping，readonly分别表示 正常，欠费停服，试用客户流量包耗尽，恶意用户，ddos攻击，无流量域名，未备案，带宽封顶，只读 示例值：normal
Area	String	加速区域，mainland，oversea或global。 示例值：mainland
Readonly	String	域名锁定状态，normal、global，分别表示未被锁定、全球锁定。 示例值：normal
Tag	Array of Tag	域名标签。 注意：此字段可能返回 null，表示取不到有效值。 示例值：null

DomainData

排序类型数据结构

被如下接口引用：DescribeEcdnDomainStatistics。

名称	类型	描述
Resource	String	域名
DetailData	Array of DetailData	结果详情

DomainDetailInfo

ECDN域名详细配置信息。

被如下接口引用：DescribeDomainsConfig。

名称	类型	描述
ResourceId	String	域名ID。
AppId	Integer	腾讯云账号ID。
Domain	String	加速域名。
Cname	String	域名CName。 注意：此字段可能返回 null，表示取不到有效值。
Status	String	域名状态，pending，rejected，processing， online，offline，deleted分别表示审核中，审核未通过，审核通过部署中，已开启，已关闭，已删除。
ProjectId	Integer	项目ID。
CreateTime	Timestamp	域名创建时间。
UpdateTime	Timestamp	域名更新时间。
Origin	Origin	源站配置。
IpFilter	IpFilter	IP黑白名单配置。 注意：此字段可能返回 null，表示取不到有效值。
IpFreqLimit	IpFreqLimit	IP限频配置。 注意：此字段可能返回 null，表示取不到有效值。
ResponseHeader	ResponseHeader	源站响应头部配置。 注意：此字段可能返回 null，表示取不到有效值。
CacheKey	CacheKey	节点缓存配置。 注意：此字段可能返回 null，表示取不到有效值。
Cache	Cache	缓存规则配置。 注意：此字段可能返回 null，表示取不到有效值。
Https	Https	Https配置。 注意：此字段可能返回 null，表示取不到有效值。
Disable	String	域名封禁状态，normal，overdue，quota，malicious，ddos，idle，unlicensed，capping，readonly分别表示 正常，欠费停服，试用客户流量包耗尽，恶意用户，ddos攻击，无流量域名，未备案，带宽封顶，只读。 注意：此字段可能返回 null，表示取不到有效值。
ForceRedirect	ForceRedirect	访问协议强制跳转配置。 注意：此字段可能返回 null，表示取不到有效值。
Area	String	加速区域，mainland，overseas或global。 注意：此字段可能返回 null，表示取不到有效值。
Readonly	String	域名锁定状态，normal、global 分别表示未被锁定，全球锁定。 注意：此字段可能返回 null，表示取不到有效值。
Tag	Array of Tag	域名标签。 注意：此字段可能返回 null，表示取不到有效值。
WebSocket	WebSocket	WebSocket配置。 注意：此字段可能返回 null，表示取不到有效值。

DomainFilter

域名查询时过滤条件。

被如下接口引用：DescribeDomains, DescribeDomainsConfig。

名称	类型	必选	描述
Name	String	是	过滤字段名，支持的列表如下： - origin：主源站。 - domain：域名。 - resourceId：域名id。 - status：域名状态，online，offline，processing。 - disable：域名封禁状态，normal，unlicensed。 - projectId：项目ID。 - fullUrlCache：全路径缓存，on或off。 - https：是否配置https，on，off或processing。 - originPullProtocol：回源协议类型，支持http，follow或https。 - area：加速区域，支持mainland，overseas或global。 - tagKey：标签键。 示例值：origin
Value	Array of String	是	过滤字段值。 示例值：[‘xxxx.test.com’]
Fuzzy	Boolean	否	是否启用模糊查询，仅支持过滤字段名为origin，domain。 示例值：false

DomainLogs

域名日志信息

被如下接口引用：DescribeEcdnDomainLogs。

名称	类型	描述
StartTime	Timestamp	日志起始时间。
EndTime	Timestamp	日志结束时间。
LogPath	String	日志下载路径。

EcdnData

访问明细数据类型

被如下接口引用：DescribeEcdnStatistics。

名称	类型	描述
Metrics	Array of String	查询指定的指标名称：Bandwidth，Flux，Request，Delay，状态码，LogBandwidth，LogFlux，LogRequest
DetailData	Array of TimestampData	明细数据组合

ForceRedirect

访问协议强制跳转配置。

被如下接口引用：DescribeDomainsConfig。

名称	类型	必选	描述
Switch	String	是	访问协议强制跳转配置开关，on或off。 注意：此字段可能返回 null，表示取不到有效值。
RedirectType	String	否	强制跳转访问协议类型，支持http，https，分别表示请求强制跳转http协议，请求强制跳转https协议。 注意：此字段可能返回 null，表示取不到有效值。
RedirectStatusCode	Integer	否	强制跳转开启时返回的http状态码，支持301或302。 注意：此字段可能返回 null，表示取不到有效值。

Hsts

HSTS 配置。

被如下接口引用：DescribeDomainsConfig。

名称	类型	必选	描述
Switch	String	是	是否开启，on或off。
MaxAge	Integer	否	MaxAge数值。 注意：此字段可能返回 null，表示取不到有效值。
IncludeSubDomains	String	否	是否包含子域名，on或off。 注意：此字段可能返回 null，表示取不到有效值。

HttpHeaderPathRule

分路径的http头部设置规则。

被如下接口引用：DescribeDomainsConfig。

名称	类型	必选	描述
HeaderMode	String	是	http头部设置方式，支持add，set或del，分别表示新增，设置或删除头部。 请求头部暂不支持set。 注意：此字段可能返回 null，表示取不到有效值。
HeaderName	String	是	http头部名称。 注意：此字段可能返回 null，表示取不到有效值。
HeaderValue	String	是	http头部值。del时可不填写该字段。 注意：此字段可能返回 null，表示取不到有效值。
RuleType	String	是	生效的url路径规则类型，支持all，file，directory或path，分别表示全部路径，文件后缀类型，目录或绝对路径生效。 注意：此字段可能返回 null，表示取不到有效值。
RulePaths	Array of String	是	url路径或文件类型列表。 注意：此字段可能返回 null，表示取不到有效值。

Https

域名https配置。

被如下接口引用：DescribeDomainsConfig。

名称	类型	必选	描述
Switch	String	是	https配置开关，on或off。开启https配置的域名在部署中状态，开关保持off。 注意：此字段可能返回 null，表示取不到有效值。
Http2	String	否	是否开启http2，on或off。 注意：此字段可能返回 null，表示取不到有效值。
OcspStapling	String	否	是否开启OCSP功能，on或off。 注意：此字段可能返回 null，表示取不到有效值。
VerifyClient	String	否	是否开启客户端证书校验功能，on或off，开启时必选上传客户端证书信息。 注意：此字段可能返回 null，表示取不到有效值。
CertInfo	ServerCert	否	服务器证书配置信息。 注意：此字段可能返回 null，表示取不到有效值。
ClientCertInfo	ClientCert	否	客户端证书配置信息。 注意：此字段可能返回 null，表示取不到有效值。
Spdy	String	否	是否开启Spdy，on或off。 注意：此字段可能返回 null，表示取不到有效值。
SslStatus	String	否	https证书部署状态，closed，deploying，deployed，failed分别表示已关闭，部署中，部署成功，部署失败。不可作为入参使用。 注意：此字段可能返回 null，表示取不到有效值。
Hsts	Hsts	否	Hsts配置 注意：此字段可能返回 null，表示取不到有效值。

IpFilter

IP黑白名单。

被如下接口引用：DescribeDomainsConfig。

名称	类型	必选	描述
Switch	String	是	IP黑白名单开关，on或off。
FilterType	String	否	IP黑白名单类型，whitelist或blacklist。 注意：此字段可能返回 null，表示取不到有效值。
Filters	Array of String	否	IP黑白名单列表。 注意：此字段可能返回 null，表示取不到有效值。

IpFreqLimit

IP限频配置。

被如下接口引用：DescribeDomainsConfig。

名称	类型	必选	描述
Switch	String	是	IP限频配置开关，on或off。
Qps	Integer	否	每秒请求数。 注意：此字段可能返回 null，表示取不到有效值。

IpStatus

节点 IP 信息

被如下接口引用：DescribeIpStatus。

名称	类型	描述
Ip	String	节点 IP
District	String	节点所属区域
Isp	String	节点所属运营商
City	String	节点所在城市
Status	String	节点状态 online：上线状态，正常调度服务中 offline：下线状态
CreateTime	Timestamp	节点 IP 添加时间

Origin

源站配置。

被如下接口引用：DescribeDomains, DescribeDomainsConfig。

名称	类型	必选	描述
Origins	Array of String	否	主源站列表，IP与域名源站不可混填。配置源站端口["origin1:port1", "origin2:port2"]，配置回源权重["origin1::weight1", "origin2::weight2"]，同时配置端口与权重 ["origin1:port1:weight1", "origin2:port2:weight2"]，权重值有效范围为0-100。 示例值：["1.1.1.1:80", "2.2.2.2:80"]
OriginType	String	否	主源站类型，支持domain，ip，分别表示域名源站，ip源站。 设置Origins时必须填写。 注意：此字段可能返回 null，表示取不到有效值。 示例值：ip
ServerName	String	否	回源时Host头部值。 注意：此字段可能返回 null，表示取不到有效值。 示例值：ecdn.test.com
OriginPullProtocol	String	否	回源协议类型，支持http，follow，https，分别表示强制http回源，协议跟随回源，https回源。 不传入的情况下默认为http回源. 注意：此字段可能返回 null，表示取不到有效值。 示例值：http
BackupOrigins	Array of String	否	备份源站列表。 示例值：["3.3.3.3:80", "4.4.4.4:80"]
BackupOriginType	String	否	备份源站类型，同OriginType。 设置BackupOrigins时必须填写。 注意：此字段可能返回 null，表示取不到有效值。 示例值：ip
AdvanceHttps	AdvanceHttps	否	HTTPS回源高级配置 注意：此字段可能返回 null，表示取不到有效值。 示例值：null

PurgeTask

刷新任务日志详情

被如下接口引用：DescribePurgeTasks。

名称	类型	描述
TaskId	String	刷新任务ID。
Url	String	刷新Url。
Status	String	刷新任务状态，fail表示失败，done表示成功，process表示刷新中。
PurgeType	String	刷新类型，url表示url刷新，path表示目录刷新。
FlushType	String	刷新资源方式，flush代表刷新更新资源，delete代表刷新全部资源。
CreateTime	Timestamp	刷新任务提交时间

ResourceData

查询对象及其对应的访问明细数据

被如下接口引用：DescribeEcdnStatistics。

名称	类型	描述
Resource	String	资源名称，根据查询条件不同分为以下几类： 具体域名：表示该域名明细数据 multiDomains：表示多域名汇总明细数据 项目 ID：指定项目查询时，显示为项目 ID all：账号维度明细数据
EcdnData	EcdnData	资源对应的数据明细

ResponseHeader

自定义响应头配置。

被如下接口引用：DescribeDomainsConfig。

名称	类型	必选	描述
Switch	String	是	自定义响应头开关，on或off。
HeaderRules	Array of HttpHeaderPathRule	否	自定义响应头规则数组。 注意：此字段可能返回 null，表示取不到有效值。

ServerCert

https服务端证书配置。

被如下接口引用：DescribeDomains, DescribeDomainsConfig。

名称	类型	必选	描述
CertId	String	否	服务器证书id，当证书为腾讯云托管证书时必填。 注意：此字段可能返回 null，表示取不到有效值。

名称	类型	必选	描述
CertName	String	否	服务器证书名称，当证书为腾讯云托管证书时必填。 注意：此字段可能返回 null，表示取不到有效值。
Certificate	String	否	服务器证书信息，上传自有证书时必填，必须包含完整的证书链信息。 注意：此字段可能返回 null，表示取不到有效值。
PrivateKey	String	否	服务器密钥信息，上传自有证书时必填。 注意：此字段可能返回 null，表示取不到有效值。
ExpireTime	Timestamp	否	证书过期时间。 注意：此字段可能返回 null，表示取不到有效值。
DeployTime	Timestamp	否	证书颁发时间。 注意：此字段可能返回 null，表示取不到有效值。
Message	String	否	证书备注信息。 注意：此字段可能返回 null，表示取不到有效值。

Sort

查询结果排序条件。

被如下接口引用：DescribeDomainsConfig。

名称	类型	必选	描述
Key	String	是	排序字段，当前支持： createTime，域名创建时间 certExpireTime，证书过期时间
Sequence	String	否	asc/desc，默认desc。

Tag

标签键和标签值

被如下接口引用：DescribeDomains, DescribeDomainsConfig。

名称	类型	必选	描述
TagKey	String	是	标签键 注意：此字段可能返回 null，表示取不到有效值。
TagValue	String	是	标签值 注意：此字段可能返回 null，表示取不到有效值。

TimestampData

时间戳与其对应的数值

被如下接口引用：DescribeEcdnStatistics。

名称	类型	描述
Time	Timestamp	数据统计时间点，采用向前汇总模式 以 5 分钟粒度为例，13:35:00 时间点代表的统计数据区间为 13:35:00 至 13:39:59
Value	Array of Float	数据值

WebSocket

WebSocket配置。

被如下接口引用：DescribeDomainsConfig。

名称	类型	必选	描述
Switch	String	是	WebSocket 超时配置开关, 开关为off时，平台仍支持WebSocket连接，此时超时时间默认为15秒，若需要调整超时时间，将开关置为on. * WebSocket 为内测功能,如需使用,请联系腾讯云工程师开白. 示例值：off
Timeout	Integer	否	设置超时时间，单位为秒，最大超时时间65秒。 注意：此字段可能返回 null，表示取不到有效值。 示例值：30

错误码

最近更新时间：2025-03-25 01:33:35

功能说明

如果返回结果中存在 Error 字段，则表示调用 API 接口失败。例如：

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

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

错误码列表

公共错误码

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

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

业务错误码

错误码	说明
FailedOperation.EcdnConfigError	域名配置更新操作失败，请重试或联系客服人员解决。
InternalError.CamSystemError	CAM鉴权错误，请稍后重试。
InternalError.EcdnConfigError	内部配置服务错误，请重试或联系客服人员解决。
InternalError.EcdnDbError	内部数据错误，请联系腾讯云工程师进一步排查。
InternalError.EcdnQuerySystemError	系统内部错误，请联系客户人员解决或稍后重试。
InternalError.EcdnSystemError	系统错误，请联系腾讯云工程师进一步排查。

错误码	说明
InternalError.Error	内部服务错误，请联系腾讯云工程师进一步排查。
InternalError.ProxyServer	后端服务错误,请稍后重试 。
InvalidParameter.EcdnCertNoCertInfo	无法解析证书信息。
InvalidParameter.EcdnConfigInvalidCache	缓存配置不合法 。
InvalidParameter.EcdnDomainInvalidStatus	域名状态不合法。
InvalidParameter.EcdnInterfaceError	内部接口错误，请联系腾讯云工程师进一步排查。
InvalidParameter.EcdnInvalidParamInterval	统计粒度不合法，请参考文档中统计分析示例。
InvalidParameter.EcdnParamError	参数错误，请参考文档中示例参数填充。
InvalidParameter.EcdnPurgeWildcardNotAllowed	刷新不支持泛域名。
InvalidParameter.EcdnStatInvalidDate	日期不合法，请参考文档中日期示例。
InvalidParameter.EcdnStatInvalidMetric	统计类型不合法，请参考文档中统计分析示例。
InvalidParameter.EcdnUrlExceedLength	URL 超过限制长度。
InvalidParameter.ParamError	参数错误。
LimitExceeded.EcdnDomainOpTooOften	域名操作过于频繁。
LimitExceeded.EcdnPurgeUrlExceedBatchLimit	刷新的Url数量超过单次限制。
LimitExceeded.EcdnPurgeUrlExceedDayLimit	刷新的Url数量超过每日限额。
ResourceNotFound.EcdnDomainNotExists	账号下无此域名，请确认后重试。
ResourceNotFound.EcdnHostNotExists	账号下无此域名，请确认后重试。
ResourceNotFound.EcdnProjectNotExists	项目不存在。
ResourceNotFound.EcdnUserNotExists	未开通ECDN服务，请开通后使用此接口。
UnauthorizedOperation.CdnAccountUnauthorized	子账号禁止查询整体数据。
UnauthorizedOperation.CdnCamUnauthorized	子账号未配置cam策略。
UnauthorizedOperation.CdnDomainUnauthorized	ECDN子账号加速域名未授权。
UnauthorizedOperation.CdnHostUnauthorized	ECDN子账号加速域名未授权。
UnauthorizedOperation.CdnNoDomainUnauthorized	子账号没有授权域名权限，请授权后重试。
UnauthorizedOperation.CdnProjectUnauthorized	子账号项目未授权。
UnauthorizedOperation.DomainNoPermission	ECDN 子账号加速域名未授权。
UnauthorizedOperation.DomainsNoPermission	ECDN 子账号加速域名未授权。
UnauthorizedOperation.EcdnAccountUnauthorized	子账号禁止查询整体数据。
UnauthorizedOperation.EcdnCamUnauthorized	子账号未配置cam策略。
UnauthorizedOperation.EcdnDomainUnauthorized	ECDN子账号加速域名未授权。
UnauthorizedOperation.EcdnHostUnauthorized	ECDN子账号加速域名未授权。

错误码	说明
UnauthorizedOperation.EcdnMigratedCdn	请前往CDN控制台进行操作 。
UnauthorizedOperation.EcdnNoDomainUnauthorized	子账号没有授权域名权限，请授权后重试。
UnauthorizedOperation.EcdnProjectUnauthorized	子账号项目未授权。
UnauthorizedOperation.EcdnUserIsSuspended	加速服务已停服，请重启加速服务后重试。
UnauthorizedOperation.EcdnUserNoWhitelist	非内测白名单用户，无该功能使用权限。
UnauthorizedOperation.NoPermission	ECDN 子账号cam未授权。
UnauthorizedOperation.OperationTooOften	操作过于频繁，请稍后重试 。
UnauthorizedOperation.ProjectNoPermission	ECDN 子账号项目未授权。
UnauthorizedOperation.ProjectsNoPermission	ECDN 子账号项目未授权。
UnauthorizedOperation.Unknown	未知错误,请稍后重试 。

全站加速网络 API 2017
简介

最近更新时间：2020-06-03 18:11:11

> **Note:**
> - 当前页面接口为旧版 API，未来可能停止维护。全站加速网络 API3.0 版本接口定义更加规范，访问时延下降显著，建议使用 全站加速网络 API3.0 。
> - 如果您需要访问全站加速网络旧版 API，可参见 API 概览 。

欢迎使用腾讯云全站加速网络（Enterprise Content Delivery Network）服务！
全站加速网络（ECDN）利用腾讯自研的最优链路算法及协议层优化，提供全球范围内稳定、安全的一站式加速服务。
请您在使用这些接口前，确保已经充分了解 ECDN 产品说明 和 计费说明 。

术语表

为了让您快速了解全站加速网络，我们对其中的一些常用术语进行了解释，如下表：

术语	中文	说明
ECDN	全站加速网络	动静融合的一站式加速服务，实现全球范围内快速、稳定、安全的数据加速传输。
CNAME	别名	域名解析中的别名记录，您可以到域名服务提供商处进行设置。
Origin	源站	用户自有的业务服务器。

快速使用入门

为了使用全站加速网络，您只需要完成以下两个步骤：

1. 添加域名您可以通过 新增加速域名 接口将域名添加入 ECDN，添加成功后，ECDN 会为您的域名分配对应的 CNAME ，您可以通过 查询域名列表 接口或登录 ECDN 控制台查看。
   - 添加的加速域名，必须尚未添加入腾讯云 CDN 或者 ECDN 服务平台；
   - 添加中国大陆区域加速的域名，必须通过工信部备案。
2. 配置 CNAME根据第一步获取的 CNAME，您需要到域名服务商处设置 CNAME ，待域名解析生效时，即开始使用全站加速网络服务。

API 概览

最近更新时间：2019-10-31 17:23:45

配置查询

接口名称	Action Name	功能描述
查询加速域名列表	GetDsaHostList	获取客户账号下所有 ECDN 加速域名的列表
查询域名配置信息	GetDsaHostInfo	根据域名 ID 获取域名配置信息
查询 HTTPS 域名列表	GetDsaHttpsHostList	获取客户账号下所有开启 HTTPS 加速功能的域名列表

内容管理

接口名称	Action Name	功能描述
缓存刷新	FlushCache	更新或删除缓存文件
刷新历史查询	GetFlushLog	查询提交的刷新 URL、刷新目录任务执行情况

域名管理

接口名称	Action Name	功能描述
创建加速域名	CreateDsaHost	添加域名至 ECDN
开启 ECDN 域名	OnlineDsaHost	根据域名 ID 启动该域名的加速服务
关闭 ECDN 域名	OfflineDsaHost	根据域名 ID 关闭该域名的加速服务
删除加速域名	DeleteDsaHost	根据域名 ID 删除该域名
修改域名配置	UpdateDsaHostInfo	根据域名 ID 修改加速配置

数据查询

接口名称	Action Name	功能描述
监控数据查询接口	GetDsaStatistics	查询监控明细数据
统计数据查询接口	GetDsaHostStatistics	查询指定时间段的统计数据值

日志接口

接口名称	Action Name	功能描述
访问日志下载链接查询接口	GetDsaHostLogs	查询 ECDN 用户访问日志下载链接

调用方式
请求结构
请求结构简介

最近更新时间：2020-03-13 16:55:34

对腾讯云的 API 接口调用是通过向腾讯云 API 的服务端地址发送请求，并按照接口说明在请求中加入相应请求参数来完成的。腾讯云 API 的请求结构由以下几部分组成：

服务地址

腾讯云 API 的服务接入地址与具体模块相关，详见各接口描述。

通信协议

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

请求方法

腾讯云 API 同时支持 POST 和 GET 请求。

> **Note:**
> 1. 不能混合使用这两种请求方式，即如果使用 GET 方式，则参数均从 Querystring 取得；如果使用 POST 方式，则参数均从 Request Body 中取得，而Querystring 中的参数将忽略。两种方式参数格式规则相同，一般情况下使用 GET，当参数字符串过长时推荐使用 POST。
> 2. 如果用户的请求方法是 GET，则对所有请求参数值均需要做 URL 编码，若为 POST，则无需对参数编码。

请求参数

腾讯云 API 的每个请求都需要指定两类参数：即公共请求参数以及接口请求参数。其中 公共请求参数 是每个接口都要用到的请求参数，而 接口请求参数 是各个接口所特有的请求参数。

字符编码

腾讯云 API 的请求及返回结果均使用 UTF-8 字符集进行编码。

公共请求参数

最近更新时间：2020-03-13 17:00:37

公共请求参数是每个接口都需要使用到的请求参数，如非必要，在各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。公共请求参数的首字母均为大写，以此区分于接口请求参数。
公共请求参数具体列表如下：

名称	类型	必选	描述
Action	String	是	具体操作的指令接口名称，例如想要调用 查询加速域名列表 接口，则 Action 参数为 GetDsaHostList。
Timestamp	UInt	是	当前 UNIX 时间戳，可记录发起 API 请求的时间。
Nonce	UInt	是	随机正整数，与 Timestamp 联合起来，用于防止重放攻击。
SecretId	String	是	在 云 API 密钥 上申请的标识身份的 SecretId，一个 SecretId 对应唯一的 SecretKey，而 SecretKey 会用来生成请求签名 Signature。具体可参考 签名方法 页面。
Signature	String	是	请求签名，用来验证此次请求的合法性，用户根据输入参数自动生成。签名生成算法可参考 签名方法 页面。

举例说明：假设用户想要查询所有接入 CDN 的域名列表，则其请求链接的形式可能如下：

```
https://dsa.api.qcloud.com/v2/index.php?
Action = GetDsaHostList
&SecretId = xxxxxxx
&Timestamp = 1465055529
&Nonce = 59485
&Signature = mysignature
&<接口请求参数>
```

一个完整的请求需要两类请求参数：公共请求参数和接口请求参数。这里只列出了上述5个公共请求参数，并未列出接口请求参数，有关接口请求参数的说明可见 接口请求参数 小节。

接口请求参数

最近更新时间：2020-03-13 17:28:03

接口请求参数与具体的接口有关，不同的接口支持的接口请求参数也不一样。接口请求参数的首字母均为小写，以此区分于公共请求参数。
以 启动 ECDN 域名 (OnlineDsaHost) 为例，其支持的接口请求参数如下：

参数名称	必选	类型	描述
hostId	是	Unsigned	域名接入 ECDN 后的标识 ID

其中各字段的说明如下：

参数名称	该接口支持的请求参数名，用户可以在使用此接口时将其作为接口请求参数。 注意：如果参数名称以“.n”结尾，则表明此参数为一个数组，使用时需要依次传入数组参数。
必选	标志此参数是否是必须输入的参数。若为“是”，则表明调用该接口必须传入此参数；若为“否”，表示可以不传入；当所有接口请求参数均不是必选时，仅使用公共请求参数就能完成正常的接口调用。
类型	此接口参数的数据类型。
描述	简要描述了此接口请求参数的内容及调用说明。

举例说明：假设用户想要启动 hostId 为 12345 的域名（hostID 可通过 域名列表查询 接口获取），则其请求链接的形式可能如下：

```
https://dsa.api.qcloud.com/v2/index.php?
&<公共请求参数>
&hostID = 12345
```

一个完整的请求需要两类请求参数：公共请求参数和接口请求参数。这里只列出了接口请求参数，并未列出公共请求参数，有关公共请求参数的说明可见 公共请求参数 小节。

最终请求形式

最近更新时间：2020-03-13 17:51:28

最终的请求 URL 由以下几部分组成：

1. 请求域名：腾讯云 API 的请求域名根据接口所属的业务模块而定，ECDN 云 API 的请求域名固定为：**dsa.api.qcloud.com**。
2. 请求路径: ECDN 云 API 的请求路径固定为 /v2/index.php。
3. 请求参数: 包括公共请求参数和接口请求参数。

最终的请求 URL 的拼接规则为：

> https:// + 请求域名 + 请求路径 + ? + 请求参数

因此，我们得到最终的请求 URL 如下，其中前5个参数为公共请求参数，后1个参数为接口请求参数。

GET 请求

```
https://dsa.api.qcloud.com/v2/index.php?
Nonce=123456789
&Timestamp=1462434006
&Action=OnlineDsaHost
&SecretId=XXXXXXXXXXXXXXXXXXXXXXXXXXXX
&Signature=XXXXXXXXXXXXXXXXXXX
&hostId=1234
```

POST 请求：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数数组如下：

```
array (
  'Nonce' => 123456789,
  'Timestamp' => 1462782282,
  'Action' => 'OnlineDsaHost',
  'SecretId' => 'XXXXXXXXXXXXXXXXXXXXXXXXXXXX',
  'Signature' => 'XXXXXXXXXXXXXXXXXXXXXXX',
  'hostId' => 1234
)
```

返回结果
正确返回结果

最近更新时间：2020-03-19 14:24:44

若 API 调用成功，则最终返回结果中的错误码 code 为 0，错误信息 message 为空，codeDesc 显示英文消息 Success，并且会显示返回的结果数据。
示例如下：

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

错误返回结果

最近更新时间：2020-03-13 17:59:08

若 API 调用失败，则最终返回结果中的错误码 code 不为0，message 字段会显示详细错误信息，codeDesc 为业务侧错误码。用户可以根据 code 在错误码页面查询相关的错误信息。
错误返回示例如下：

```
{
    "code": 4000,
    "message": "(2000) 加速域名不存在",
    "codeDesc": "InvalidParameter"
}
```

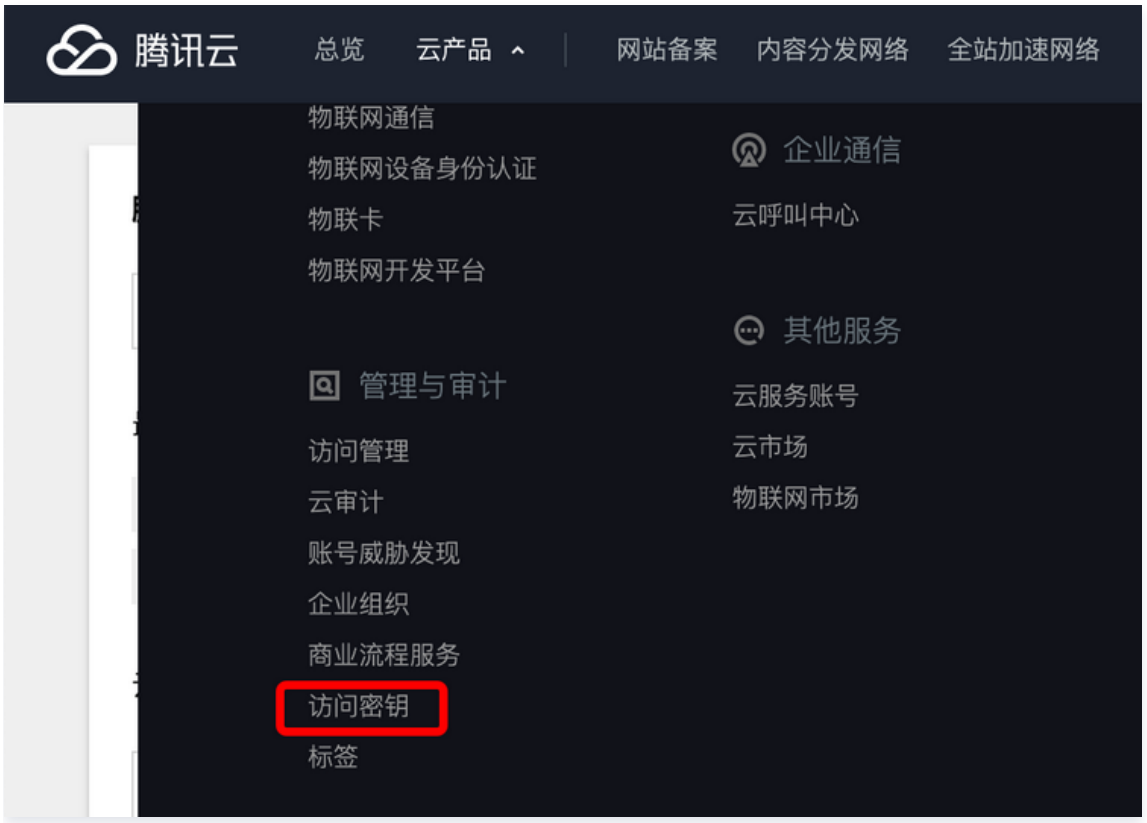
签名方法
签名方法

最近更新时间：2024-11-08 11:06:31

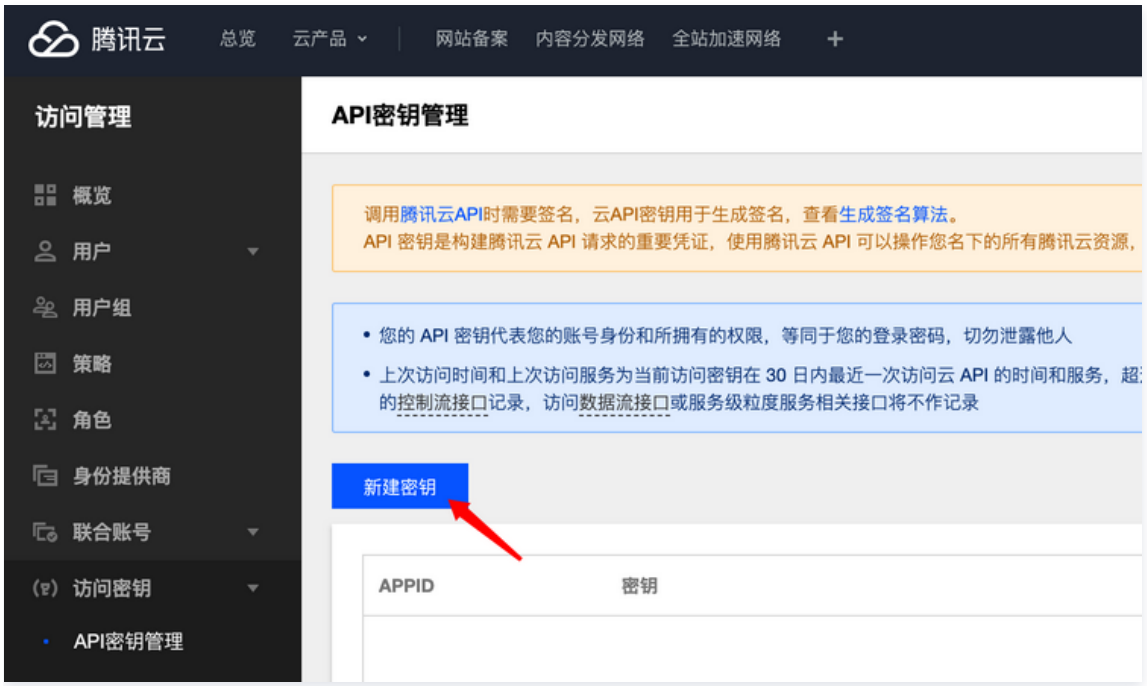
腾讯云 API 会对每个访问的请求进行身份验证，即每个请求都需要在公共请求参数中包含签名信息（Signature）以验证用户身份。签名信息由用户所执有的安全凭证生成，安全凭证包括 SecretId 和 SecretKey，若用户还没有安全凭证，则需要在腾讯云官网上自主申请，否则就无法调用云 API 接口。

申请安全凭证

在第一次使用云 API 之前，用户需要在腾讯云控制台上申请安全凭证。安全凭证包括 SecretId 和 SecretKey，其中 SecretId 是用于标识 API 调用者身份的，而SecretKey 是用于加密签名字符串和服务器端验证签名字符串的密钥。用户应严格保管其 SecretKey，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）单击【云产品】，选择【管理与审计】栏下的【访问密钥】，进入云 API 密钥管理页面。



3）单击【新建密钥】，即可创建一对 SecretId / SecretKey，每个账号最多可以拥有两对 SecretId / SecretKey。



生成签名串

> **Note:**
> - 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全性。
> - 以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见：https://cloud.tencent.com/document/product/1278/85305。

有了安全凭证 SecretId 和 SecretKey后，就可以生成签名串了。下面给出了一个生成签名串的详细过程。
假设用户的 SecretId 和 SecretKey 分别是：

```
SecretId: ****************************************
SecretKey: ******************************
```

> **Note:**
> 这里只是示例，请用户根据自己实际的 SecretId 和 SecretKey 进行后续操作。

以 查询加速域名列表（GetDsaHostList）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	GetDsaHostList
SecretId	密钥 Id	**
Timestamp	当前时间戳	1463122059
Nonce	随机正整数	13029
offset	偏移量	0
length	查询输出数目	10

由上表可以看出，请求参数中的公共请求参数只有4个：Action、SecretId、Timestamp、Nonce，而不是在“公共请求参数”中所述的5个，而第5个参数 Signature（签名串）正是由其他参数（包括指令请求参数）共同生成的，具体步骤如下：

1. 对参数排序

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

```
{
    "Action": "GetDsaHostList",
    "Nonce": 13029,
    "SecretId": "****************************************",
    "Timestamp": 1463122059,
    "length": 10,
    "offset": 0
}
```

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

2. 拼接请求字符串

此步骤生成请求字符串。
将把上一步排序好的请求参数格式化成“参数名称”=“参数值”的形式，如对 Action 参数，其参数名称为 "Action"，参数值为 "GetDsaHostList"，因此格式化后就为 Action=GetDsaHostList。

> **Note:**
> 1. “参数值”为原始值而非 URL 编码后的值。
> 2. 若输入参数中包含下划线，则需要将其转换为“.”。

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=GetDsaHostList&Nonce=13029&SecretId=****************************************&Timestamp=14631
22059&length=10&offset=0
```

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：查看域名信息(GetDsaHostList)的请求域名为：dsa.api.qcloud.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：云 API 的请求路径固定为 /v2/index.php。
4. 请求字符串：即上一步生成的请求字符串。

签名原文串的拼接规则为：

```
请求方法 + 请求主机 +请求路径 + ? + 请求字符串
```

若为 GET 请求，示例的拼接结果为：

```
GETdsa.api.qcloud.com/v2/index.php?
Action=GetDsaHostList&Nonce=13029&SecretId=****************************************&Timestamp=14631
22059&length=10&offset=0
```

若为 POST 请求，示例的拼接结果为：

```
POSTdsa.api.qcloud.com/v2/index.php?
Action=GetDsaHostList&Nonce=13029&SecretId=****************************************&Timestamp=14631
22059&offset=0
```

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 = 'pxPgRWD******qBTDk7WmeRZSmPco0';
$srcStr = 'GETdsa.api.qcloud.com/v2/index.php?
Action=GetDsaHostList&Nonce=13029&SecretId=****************************************&Timestamp=146312
2059&length=10&offset=0';
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));
echo $signStr;
```

最终得到的签名串为：

```
yvImfESYa0C1WMcHTX+KuA2BFOs=
```

若为 POST 请求，则最终的签名结果为：

```
uFT/BG266+TprJIWb5G7tt5gtyI=
```

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。

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。

> **Note:**
> 1. 如果用户的请求方法是 GET，则对所有请求参数值均需要做 URL 编码。
> 2. 部分语言库会自动对 URL 进行编码，重复编码会导致签名校验失败。

如上一步生成的签名串为：

```
bWMMAR1eFGjZ5KWbfxTlBiLiNLc=
```

则其编码后为：

```
yvImfESYa0C1WMcHTX%2BKuA2BFOs%3D
```

因此，最终得到的签名串请求参数(Signature)为：`yvImfESYa0C1WMcHTX%2BKuA2BFOs%3D`，它将用于生成最终的请求 URL。

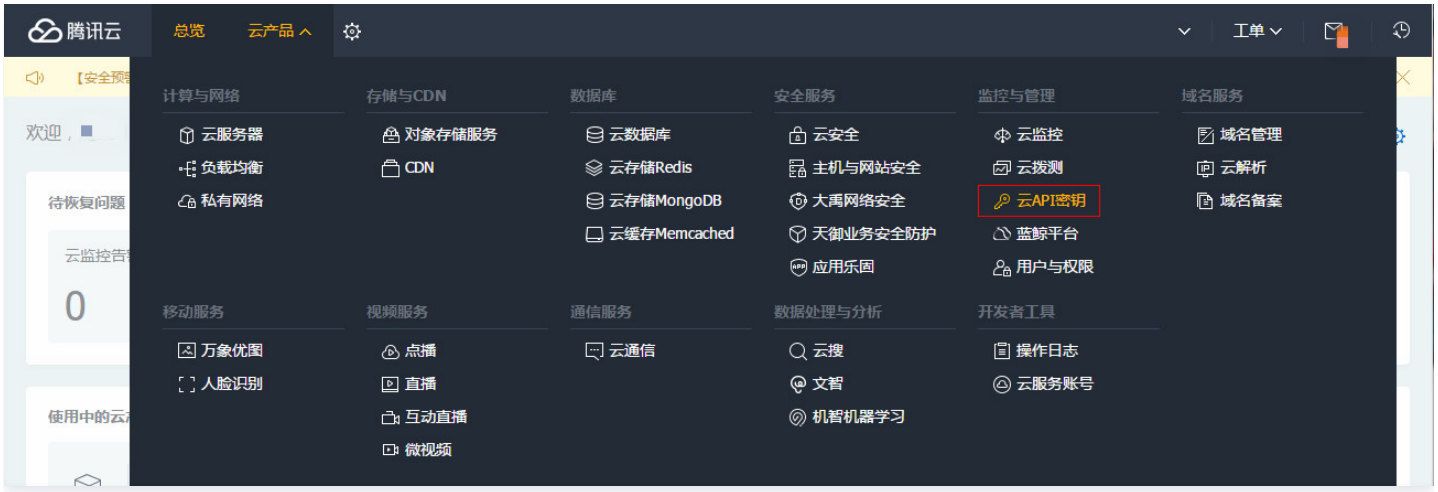
签名方法（SHA256）

最近更新时间：2024-11-08 11:06:39

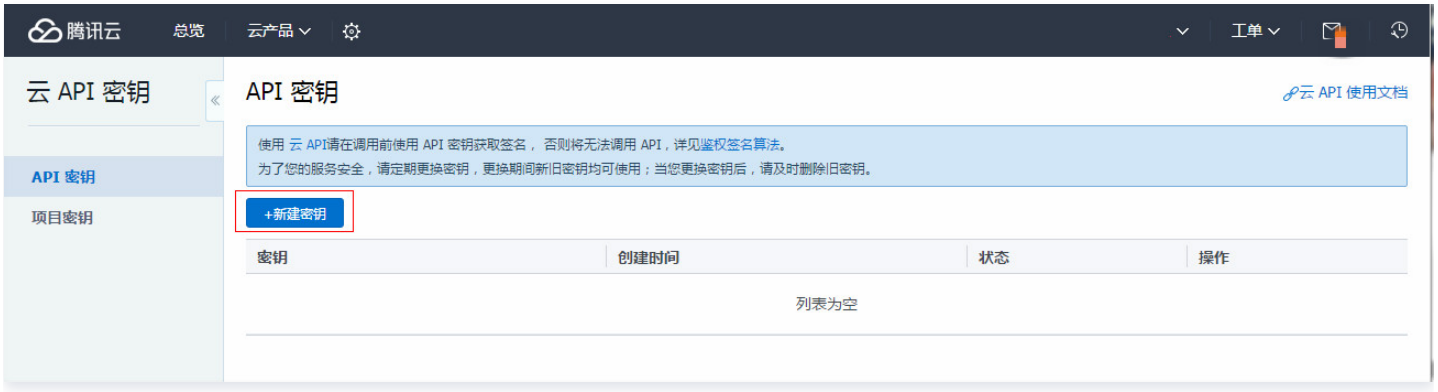
腾讯云 API 会对每个访问的请求进行身份验证，即每个请求都需要在公共请求参数中包含签名信息（Signature）以验证用户身份信息和请求参数。签名信息由用户所执有的安全凭证生成，安全凭证包括 SecretId 和 SecretKey，若用户还没有安全凭证，则需要在腾讯云官网上自主申请，否则就无法调用云API接口。

申请安全凭证

在第一次使用云 API 之前，用户需要在腾讯云控制台上申请安全凭证。安全凭证包括 SecretId 和 SecretKey，其中 SecretId 是用于标识 API 调用者身份的，而SecretKey 是用于加密签名字符串和服务器端验证签名字符串的密钥。用户应严格保管其 SecretKey，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）单击【云产品】，选择【监控与管理】栏下的【云 API 密钥】，进入云 API 密钥管理页面。



3）单击【新建密钥】，即可创建一对 SecretId/SecretKey，每个账号最多可以拥有两对 SecretId/SecretKey。



生成签名串

> **Note:**
> - 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全性。
> - 以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见：https://cloud.tencent.com/document/product/1278/85305。

有了安全凭证 SecretId 和 SecretKey 后，就可以生成签名串了。下面给出了一个生成签名串的详细过程。
假设用户的 SecretId 和 SecretKey 分别是：

```
SecretId: ****************************************
SecretKey: ********************************
```

> **Note:**
> 这里只是示例，请用户根据自己实际的 SecretId 和 SecretKey 进行后续操作。
> 以 查询加速域名列表(GetDsaHostList) 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	GetDsaHostList
SecretId	密钥 Id	************************************
Timestamp	当前时间戳	1463122059
Nonce	随机正整数	13029
SignatureMethod	签名方式	默认填充字符串：HmacSHA256
offset	偏移量	0
length	查询输出数目	10

由上表可以看出，请求参数中的公共请求参数只有 4 个：Action、SecretId、Timestamp、Nonce、SignatureMethod（SHA256 专用），而不是在“公共请求参数”中所述的5个，而第 5 个参数 Signature（签名串）正是由其他参数（包括指令请求参数）共同生成的，具体步骤如下:

1. 对参数排序

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

```
'Action' : 'GetDsaHostList',
'Nonce' : 48059,
'SecretId' : '************************************',
'Timestamp' : 1502197934,
'SignatureMethod':'HmacSHA256',
'length' : 10,
'offset' : 0
```

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

2. 拼接请求字符串

此步骤生成请求字符串。
将把上一步排序好的请求参数格式化成“参数名称”=“参数值”的形式，如对Action参数，其参数名称为"Action"，参数值为"GetDsaHostList"，因此格式化后就为Action=GetDsaHostList。

> **Note:**
> 1. “参数值”为原始值而非 url 编码后的值。
> 2. 若输入参数中包含下划线，则需要将其转换为“.”。

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=GetDsaHostList
&Nonce=48059
&SecretId=************************************
&SignatureMethod=HmacSHA256
&Timestamp=1502197934
&length=10
```

```
&offset=0
```

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：查看域名信息(GetDsaHostList)的请求域名为：dsa.api.qcloud.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：云 API 的请求路径固定为 /v2/index.php。
4. 请求字符串：即上一步生成的请求字符串。

签名原文串的拼接规则为：

> 请求方法 + 请求主机 +请求路径 + ? + 请求字符串

若为 GET 请求，示例的拼接结果为：

```
GETdsa.api.qcloud.com/v2/index.php?
Action=GetDsaHostList
&Nonce=48059
&SecretId=****************************************
&SignatureMethod=HmacSHA256
&Timestamp=1502197934
&length=10
&offset=0
```

若为 POST 请求，示例的拼接结果为：

```
POSTdsa.api.qcloud.com/v2/index.php?
Action=GetDsaHostList
&Nonce=48059
&SecretId=****************************************
&SignatureMethod=HmacSHA256
&Timestamp=1502197934
&length=10
&offset=0
```

4. 生成签名串

此步骤生成签名串。
首先使用 HMAC-SHA256 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 = 'pxPgRWD******qBTDk7WmeRZSmPco0';
$srcStr = 'GETdsa.api.qcloud.com/v2/index.php?
Action=GetDsaHostList&Nonce=48059&SecretId=****************************************&SignatureMethod=
HmacSHA256&Timestamp=1502197934&length=10&offset=0';
$signStr = base64_encode(hash_hmac('sha256', $srcStr, $secretKey, true));
echo $signStr;
```

最终得到的签名串为：

```
oC20lImZgsEZYZqHYQnbvBxEkIFUxgoDhE3GkQA8Ax8=
```

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。

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。
注意：如果用户的请求方法是 GET，则对所有请求参数值均需要做 URL 编码。
如上一步生成的签名串为：

```
oC20llmZgsEZYZqHYQnbvBxEklFUxgoDhE3GkQA8Ax8=
```

则其编码后为：

```
oC20llmZgsEZYZqHYQnbvBxEklFUxgoDhE3GkQA8Ax8%3D
```

因此，最终得到的签名串请求参数(Signature)为：oC20llmZgsEZYZqHYQnbvBxEklFUxgoDhE3GkQA8Ax8%3D，它将用于生成最终的请求 URL。

> **Note:**
> 部分语言库会自动对 URL 进行编码，重复编码会导致签名校验失败。

签名示例

最近更新时间：2020-05-26 15:45:49

示例代码

示例代码下载

PHP: 前往Github下载
示例代码仅供参考， 请根据实际情况使用。

示例代码（PHP）

> **Note:**
> - 建议用户使用子账号密钥 + 环境变量的方式调用 SDK，提高 SDK 使用的安全性。为子账号授权时，请遵循 最小权限指引原则，防止泄漏目标存储桶或对象之外的资源。
> - 如果您一定要使用永久密钥，建议遵循 最小权限指引原则 对永久密钥的权限范围进行限制。

以 DescribeCdnHosts 为例：

```
<?php

/*DSA API请求域名*/
$HttpUrl="dsa.api.qcloud.com";

/*除非有特殊说明，其它接口都支持GET及POST*/
$HttpMethod="GET";

/*是否https协议，DSA 云API接口都必须为https协议*/
$isHttps =true;

/*需要填写你的密钥，可从  https://console.cloud.tencent.com/capi 获取 SecretId 及 $secretKey*/
$secretId = getenv('CDN_SECRET_ID');     // 用户的 SecretId，建议使用子账号密钥，授权遵循最小权限指引，降低使用风险。子账号密钥获取可参考https://cloud.tencent.com/document/product/598/37140
$secretKey = getenv('CDN_SECRET_KEY');   // 用户的 SecretKey，建议使用子账号密钥，授权遵循最小权限指引，降低使用风险。子账号密钥获取可参考https://cloud.tencent.com/document/product/598/37140

/*调用接口的名称*/
$action='GetDsaHostList';

/*下面为接口的公共请求参数*/
$COMMON_PARAMS = array(
        'Nonce'=> rand(),
        'Timestamp'=>time(NULL),
        'Action'=>$action,
        'SecretId'=> $SecretId,
);

/*下面为接口的接口请求参数 */
$PRIVATE_PARAMS = array(
    'offset' => 0,
    'length' => 10,
);
```

```
/**************************************************************************/

CreateRequest($HttpUrl,$HttpMethod,$COMMON_PARAMS,$secretKey, $PRIVATE_PARAMS, $isHttps);

function CreateRequest($HttpUrl,$HttpMethod,$COMMON_PARAMS,$secretKey, $PRIVATE_PARAMS, $isHttps)
{
  $FullHttpUrl = $HttpUrl."/v2/index.php";

    /***************对请求参数 按参数名 做字典序升序排列，注意此排序区分大小写*************/
    $ReqParaArray = array_merge($COMMON_PARAMS, $PRIVATE_PARAMS);
    ksort($ReqParaArray);

    /*********************************生成签名原文*********************************
     * 将 请求方法， URI地址，及排序好的请求参数 按照下面格式 拼接在一起， 生成签名原文，此请求中的原文为
     * GETdsa.api.qcloud.com/v2/index.php?Action=GetDsaHostList&Nonce=13029
     * &SecretId=***************************************&Timestamp=1463122059&length=10&offset=0
     * ************************************************************************/
    $SigTxt = $HttpMethod.$FullHttpUrl."?";

    $isFirst = true;
    foreach ($ReqParaArray as $key => $value)
    {
        if (!$isFirst)
        {
            $SigTxt = $SigTxt."&";
        }
        $isFirst= false;

        /*拼接签名原文时，如果参数名称中携带_，需要替换成.*/
        if(strpos($key, '_'))
        {
            $key = str_replace('_', '.', $key);
        }

        $SigTxt=$SigTxt.$key."=".$value;
    }

    /*********************根据签名原文字符串 $SigTxt，生成签名 Signature******************/
    $Signature = base64_encode(hash_hmac('sha1', $SigTxt, $secretKey, true));
```

```
/*拼接请求串,对于请求参数及签名，需要进行urlencode编码*****/
$Req = "Signature=".urlencode($Signature);
foreach ($ReqParaArray as $key => $value)
{
  $Req=$Req."&".$key."=".urlencode($value);
}
```

```
    /*********************************发送请求*********************************/
    if($HttpMethod === 'GET')
    {
        if($isHttps === true)
        {
            $Req="https://".$FullHttpUrl."?".$Req;
```

```
        }
        else
        {
            $Req="http://".$FullHttpUrl."?".$Req;
        }

        $Rsp = file_get_contents($Req);

    }
    else
    {
        if($isHttps === true)
        {
            $Rsp= SendPost("https://".$FullHttpUrl,$Req,$isHttps);
        }
        else
        {
            $Rsp= SendPost("http://".$FullHttpUrl,$Req,$isHttps);
        }
    }

    var_export(json_decode($Rsp,true));
}

function SendPost($FullHttpUrl,$Req,$isHttps)
{

        $ch = curl_init();
        curl_setopt($ch, CURLOPT_POST, 1);
        curl_setopt($ch, CURLOPT_POSTFIELDS, $Req);

        curl_setopt($ch, CURLOPT_URL, $FullHttpUrl);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
        if ($isHttps === true) {
            curl_setopt($ch, CURLOPT_SSL_VERIFYPEER,  false);
            curl_setopt($ch, CURLOPT_SSL_VERIFYHOST,  false);
        }

        $result = curl_exec($ch);

        return $result;
}
?>
```

配置查询相关接口
加速域名列表查询

最近更新时间：2019-12-03 10:35:02

接口描述

本接口（ GetDsaHostList ）用于查询账号下所有 ECDN 加速域名的列表。
接口请求域名： `dsa.api.qcloud.com`

> ⚠ 注意：
> 该接口调用频次上限为100次/分钟，超出上限则会返回错误，请勿高频调用。

输入参数

以下请求参数列表仅列出了接口请求参数，正式调用时需要加上 公共请求参数，见公共请求参数页面。其中，此接口的 Action 字段为 GetDsaHostList。

参数名称	必选	类型	描述
offset	否	Unsigned	偏移长度，默认0，表示不偏移。
length	否	Unsigned	最大长度，默认3000，表示最多返回3000个域名信息。
includeDeleted	否	Int	是否包括已删除域名，默认0，表示不包括删除域名。

> ⚠ 注意：
> - 该接口每次最多可查询3000个域名信息。
> - 当查询的域名较多时，可以利用 offset 和 length 进行分页查询。

输出参数

参数名称	类型	描述
code	Int	公共错误码，0表示成功，其他值表示失败。详见错误码页面的 公共错误码。
message	String	模块错误信息描述，与接口相关。
codeDesc	String	英文错误信息，或业务侧错误码。
data	Object	结果数据，详细说明见下文 **data 字段说明**。

data 字段说明

参数名称	类型	描述
hosts	Object	域名详细信息，详细字段见下文 **hosts 字段说明**。
total	Int	账户下域名总数。

hosts 字段说明

参数名称	类型	描述

host_id	Int	域名接入 ECDN 后的标识 ID。
app_id	Int	域名归属人的 AppID。
project_id	Unsigned	域名所属的项目 ID。
host	String	加速域名。
cname	String	域名接入 ECDN 的 CNAME 别名。
status	String	域名当前状态，online：启用，offline：关闭。
progress	Int	配置部署状态，0：表示部署完毕，1：表示正在部署中。
mode	Int	域名封禁状态，0：表示域名未被封禁。
locked	Int	运维锁状态，0：表示未锁定，用户可直接修改配置，1：表示锁定状态，需要提交工单修改配置。
deleted	Int	配置存在状态，0：表示配置未删除，1：表示配置已删除。
message	String	信息描述。

代码示例

配置需求示例

获取该账户下所有域名列表。

GET 请求

```
https://dsa.api.qcloud.com/v2/index.php?
Action=GetDsaHostList
&SecretId=XXXXXXXXXXXXXXXXXXXXXXXX
&Timestamp=1462440051
&Nonce=123456789
&Signature=XXXXXXXXXXXXXXXXXXXXXXXX
&offset=0
&length=2048
&includeDeleted=1
```

POST请求

POST请求时，参数填充在HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array (
  'Action' => 'GetDsaHostList',
  'SecretId' => 'SecretId',
  'Timestamp' => 1462782282,
  'Nonce' => 123456789,
  'Signature' => 'Signature',
  'offset' => '0',
  'length' => '2048',
  'includeDeleted' => '1'
)
```

返回结果示例

查询成功


```
{
    "code": 0,
    "message": "",
    "codeDesc": "Success",
    "data": {
        "hosts": [
            {
                 "host_id": "******",
                "app_id": "******",
                "project_id": 0,
                "host": "dsatest.qcloud.com",
                "cname": "dsatest.qcloud.com.dsa.dnsv1.com",
                "status": "online",
                "deleted": 0,
                "message": ""
            },
            {
                "host_id": "******",
                "app_id": "******",
                "project_id": 0,
                "host": "dsatest2.qcloud.com",
                "cname": "dsatest2.qcloud.com.dsa.dnsv1.com",
                "status": "offline",
                "deleted": 0,
                "message": ""
            }
        ],
        "total": 2
    }
}
```


查询失败


```
{
    "code": 4100,
    "message": "鉴权失败，请参考文档中鉴权部分。",
    "codeDesc": "AuthFailure"
}
```

域名列表查询接口

最近更新时间：2021-01-22 10:51:24

接口描述

本接口（ GetDsaHostList ）用于查询账号下所有 ECDN 加速域名的列表。接口请求域名： `dsa.api.qcloud.com`

> **Note:**
> 该接口调用频次上限为100次/分钟，超出上限则会返回错误，请勿高频调用。

输入参数

以下请求参数列表仅列出了接口请求参数，正式调用时需要加上 公共请求参数 ，见公共请求参数页面。其中，此接口的 Action 字段为 GetDsaHostList。

参数名称	必选	类型	描述
offset	否	Unsigned	偏移长度，默认0，表示不偏移。
length	否	Unsigned	最大长度，默认3000，表示最多返回3000个域名信息。
includeDeleted	否	Int	是否包括已删除域名，默认0，表示不包括删除域名。

> **Note:**
> - 该接口每次最多可查询3000个域名信息。
> - 当查询的域名较多时，可以利用 offset 和 length 进行分页查询。

输出参数

参数名称	类型	描述
code	Int	公共错误码，0表示成功，其他值表示失败。详见错误码页面的 公共错误码 。
message	String	模块错误信息描述，与接口相关。
codeDesc	String	英文错误信息，或业务侧错误码。
data	Object	结果数据，详细说明见下文 **data 字段说明**。

data 字段说明

参数名称	类型	描述
hosts	Object	域名详细信息，详细字段见下文 **hosts 字段说明**。
total	Int	账户下域名总数。

hosts 字段说明

参数名称	类型	描述
host_id	Int	域名接入 ECDN 后的标识 ID。
app_id	Int	域名归属人的 AppID。
project_id	Unsigned	域名所属的项目 ID。

host	String	加速域名。
cname	String	域名接入 ECDN 的 CNAME 别名。
status	String	域名当前状态，online：启用，offline：关闭。
progress	Int	配置部署状态，0：表示部署完毕，1：表示正在部署中。
mode	Int	域名封禁状态，0：表示域名未被封禁。
locked	Int	运维锁状态，0：表示未锁定，用户可直接修改配置，1：表示锁定状态，需要提交工单修改配置。
deleted	Int	配置存在状态，0：表示配置未删除，1：表示配置已删除。
message	String	信息描述。

代码示例

配置需求示例

获取该账户下所有域名列表。

GET 请求

```
https://dsa.api.qcloud.com/v2/index.php?
Action=GetDsaHostList
&SecretId=XXXXXXXXXXXXXXXXXXXXXXXX
&Timestamp=1462440051
&Nonce=123456789
&Signature=XXXXXXXXXXXXXXXXXXXXXXXX
&offset=0
&length=2048
&includeDeleted=1
```

POST请求

POST请求时，参数填充在HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array (
  'Action' => 'GetDsaHostList',
  'SecretId' => 'SecretId',
  'Timestamp' => 1462782282,
  'Nonce' => 123456789,
  'Signature' => 'Signature',
  'offset' => '0',
  'length' => '2048',
  'includeDeleted' => '1'
)
```

返回结果示例

查询成功

```
{
    "code": 0,
    "message": "",
    "codeDesc": "Success",
    "data": {
        "hosts": [
            {
                 "host_id": "******",
                "app_id": "******",
                "project_id": 0,
                "host": "dsatest.qcloud.com",
                "cname": "dsatest.qcloud.com.dsa.dnsv1.com",
                "status": "online",
                "deleted": 0,
                "message": ""
            },
            {
                "host_id": "******",
                "app_id": "******",
                "project_id": 0,
                "host": "dsatest2.qcloud.com",
                "cname": "dsatest2.qcloud.com.dsa.dnsv1.com",
                "status": "offline",
                "deleted": 0,
                "message": ""
            }
        ],
        "total": 2
    }
}
```

查询失败

```
{
    "code": 4100,
    "message": "鉴权失败，请参考文档中鉴权部分。",
    "codeDesc": "AuthFailure"
}
```

Https域名列表查询接口

最近更新时间：2021-01-22 10:50:02

接口描述

本接口（ GetDsaHttpsHostList ）用于查询账号下所有开启 HTTPS 加速功能的域名列表。接口请求域名： `dsa.api.qcloud.com`

> **Note:**
> 该接口调用频次上限为100次/分钟，超出上限则会返回错误，请勿高频调用。

输入参数

以下请求参数列表仅列出了接口请求参数，正式调用时需要加上公共请求参数，详情请参见 公共请求参数 页面。其中，此接口的 Action 字段为 `GetDsaHttpsHostList` 。

参数	必选	类型	描述
offset	否	Unsigned	偏移长度，默认0，表示不偏移
length	否	Unsigned	最大长度，默认3000，表示最多返回3000个域名信息

> **Note:**
> 1. 该接口每次最多可查询3000个域名信息。
> 2. 当查询的域名较多时，可以利用 offset 和 length 进行分页查询。

输出参数

参数	类型	描述
code	Int	公共错误码，0表示成功，其他值表示失败。详情请参见错误码页面的 公共错误码
message	String	模块错误信息描述，与接口相关
codeDesc	String	英文错误信息，或业务侧错误码
data	Object	结果数据，详细说明见下文

data 字段说明

参数	类型	描述
hosts	array	域名 HTTPS 配置信息，详情请参见下文 hosts 字段说明
total	Unsigned	开启 HTTPS 加速功能的域名总数量

hosts 字段说明

参数	类型	描述
host_id	Int	域名接入 ECDN 后的标识 ID
host	String	加速域名
Https	Object	HTTPS 配置信息，详情请参见下文 HTTPS 字段说明

HTTPS 字段说明

参数	类型	描述
type	Unsigned	HTTPS 开关 0：表示关闭 2：表示启用，并且采用 HTTP 回源 4：表示启用，并采用协议跟随回源
cert_id	String	腾讯云托管证书 ID，当使用腾讯云托管证书时返回此参数
cn	String	证书域名
expire	String	SSL 证书过期时间
message	String	证书备注信息
http2	String	HTTP 2.0开关 on：表示开启 off：表示关闭
redirection	String	HTTPS 强制跳转开关 on：表示开启 off：表示关闭

Note:
当开启 HTTPS 强制跳转功能时，ECDN 加速节点会强迫客户端将 HTTP 请求重定向成 HTTPS 请求。

示例代码

配置需求示例

获取该账户下所有开启 HTTPS 加速功能的域名列表。

GET 请求

```
https://dsa.api.qcloud.com/v2/index.php?
Action=GetDsaHttpsHostList
&SecretId=XXXXXXXXXXXXXXXXXXXXXXXX
&Timestamp=1462440051
&Nonce=123456789
&Signature=XXXXXXXXXXXXXXXXXXXXXXXX
```

POST 请求

POST 请求时，参数填充在 HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array (
  'Action' => 'GetDsaHttpsHostList',
  'SecretId' => 'SecretId',
  'Timestamp' => 1462782282,
  'Nonce' => 123456789,
  'Signature' => 'Signature'
)
```

返回结果示例

查询成功

```
{
```

```
    "code": 0,
    "message": "",
    "codeDesc": "Success",
    "data": {
        "hosts": [{
                "host_id": "******",
                "host": "dsatest1.qcloud.com",
                "https": {
                    "type": "2",
                    "cert_id": "K9FAhubv",
                    "message": "",
                    "cn": "*.qcloud.com",
                    "expire": "2018-08-24 07:59:59"
                }
            },
            {
                "host_id": "******",
                "host": "dsatest2.qcloud.com",
                "https": {
                    "type": "4",
                    "cert_id": "K9FAhubv",
                    "message": "",
                    "cn": "*.qcloud.com",
                    "expire": "2018-08-24 07:59:59"
                }
            },
            {
                "host_id": "******",
                "host": "dsatest3.elliotxing.com",
                "https": {
                    "type": "4",
                    "message": "",
                    "cn": "*.elliotxing.com",
                    "expire": "2018-08-24 07:59:59"
                }
            }
        ],
        "total": 3
    }
}
```


查询失败


```
{
    "code": 4100,
    "message": "鉴权失败，请参考文档中鉴权部分。",
    "codeDesc": "AuthFailure"
}
```

内容管理相关接口
缓存刷新

最近更新时间：2019-10-31 17:23:14

接口描述

本接口（FlushCache）更新或删除缓存文件。
请求域名：`dsa.api.qcloud.com`

可支持配置

- 刷新 URL
- 刷新目录

> **Note:**
> 该接口调用频次上限为100次/分钟，超出上限则会返回错误，请勿高频调用

输入参数

以下请求参数列表仅列出了接口请求参数，正式调用时需要加上公共请求参数，详情请参见 公共请求参数 。
其中，此接口的 Action 字段为 FlushCache。

参数名称	是否必选	类型	描述
urlList	是	String	刷新 URL /目录列表（JSON）。 例如 `["http://www.abc.com/","https://www.mmm.com/"]`
flushType	是	Int	刷新类型。 0：URL 刷新 1：目录刷新变更资源，会比对节点内容与源站内容的 Last-Modify，若未变化，则不会进行资源刷新 2：目录刷新所有资源，会直接删除目录下所有资源，此操作会导致源站压力增大，请谨慎操作

> **Note:**
> 请采用 POST 方式提交，防止出现 URL 长度越界问题。

输出参数

参数名称	类型	描述
code	Int	公共错误码，0表示成功，其他值表示失败。 详见错误码页面 公共错误码
message	String	模块错误信息描述，与接口相关。
codeDesc	String	英文错误信息，或业务侧错误码。 详见错误码页面 业务错误码
data	Array	返回结果数据，详情见下方 Data 详细说明

Data 详细说明

参数名称	类型	描述
count	Int	此次刷新提交的 URL /目录数目
task_id	String	此次刷新任务对应的 ID

调用示例

GET 请求

由于 URL 长度限制，请不要采用 Get 方式提交请求。

POST 请求

POST 请求时，参数填充在 HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array (
  'Action' => ' FlushCache ',
   'SecretId' => 'SecretId',
   'Timestamp' => 1462782282,
   'Nonce' => 123456789,
   'Signature' => 'Signature',
  ' urlList ' => ' ["http://www.abc.com/", "https://www.mmm.com/"]',
  ' flushType ' => 1
)
```

结果示例

```
{
    "code": 0,
    "message": "",
    "codeDesc": "Success",
    "data": {
             "count": 2,
               " task_id ": “xxxxxxxxxxxxxx”
    }
}
```

刷新历史查询

最近更新时间：2019-10-31 17:23:25

接口描述

本接口（GetFlushLog）查询提交的刷新 URL、刷新目录任务执行情况。
请求域名：`dsa.api.qcloud.com`

可支持配置

- 查询提交的刷新 URL 任务执行情况
- 查询提交的刷新目录任务执行情况

> **Note:**
> 该接口调用频次上限为100次/分钟，超出上限则会返回错误，请勿高频调用

输入参数

以下请求参数列表仅列出了接口请求参数，正式调用时需要加上公共请求参数，详情请参见 公共请求参数 。
其中，此接口的 Action 字段为 GetFlushLog。

参数名称	是否必选	类型	描述
type	是	Int	刷新类型。 0：URL刷新 1：目录刷新
date	否	String	指定查询的日期。例：2019-08-12
taskId	否	String	指定查询的任务 ID
keyword	否	String	查询关键字
status	否	Int	任务状态。 -1：失败 1：刷新中 2：完成
offset	否	Int	查询起始偏移量，从0开始。传入10表示从第10条开始
limit	否	Int	查询限制条数

输出参数

参数名称	类型	描述
total	Int	总数，用于分页
logs	Log 数组	刷新记录

Log 定义

参数名称	类型	描述
task_id	String	任务 ID
url	String	刷新 URL
type	Int	刷新类型。 0：URL 刷新 1：目录刷新变更资源 2：目录刷新所有资源
datetime	String	刷新时间
status	Int	任务状态。 -1：失败 1：刷新中 2：完成

调用示例

查询2019-9-20日 URL 刷新记录

GET 请求

GET 请求需要将所有参数都加在 URL 后：

```
https://dsa.api.qcloud.com/v2/index.php?
Action=GetFlushLog
&SecretId=xxxxxxxxxxxxxxxxxxxxxxxx
&Nonce=44
&Timestamp=1572349397
&date=2019-09-20
&type=0
&Signature=xxxxxxxxxxxxxxxxxxxxx
```

POST 请求

POST 请求时，参数填充在 HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array (
'Action' => 'GetFlushLog',
'SecretId' => 'xxxxxxxxxxxxxxxxxxxxxxxx',
'Nonce' => 44,
'Timestamp' => 1572349397,
'Signature'=>'xxxxxxxxxxxxxxxxxxxxx',
'date' =>'2019-09-20',
'type'=> 0
)
```

结果示例

```
{
    "code": 0,
    "message": "",
    "codeDesc": "Success",
    "data": {
        "total": 2,
        "logs": [
            {
                "task_id": "1572347900707791519",
                "url": "http://xxxxxxxxxxxxx/1.jpg",
                "type": 0,
                "datetime": "2019-09-20 19:18:20",
                "status": 2
            },
            {
                "task_id": "1572347899565885803",
                "url": "http://xxxxxxxxxxxxxxxx /2.jpg",
                "type": 0,
                "datetime": "2019-09-20 19:18:19",
```

```
                "status": 2
            }
        ]
    }
}
```

域名管理相关接口
新增加速域名

最近更新时间：2021-01-22 10:56:14

接口描述

本接口（ CreateDsaHost ）用于新增 ECDN 加速域名。
接口请求域名： `dsa.api.qcloud.com`

- 接入域名必须尚未接入腾讯云 CDN 或 ECDN 服务。
- 接入的域名必须通过工信部备案。
- 使用此接口，一次仅能添加一个域名至腾讯云 ECDN。
- 使用此接口新建的域名默认处于启用状态。
- 该接口调用频次上限为100次/分钟，超出上限则会返回错误，请勿高频调用。

输入参数

以下请求参数列表仅列出了接口请求参数，正式调用时需要加上公共请求参数，详情请参见 公共请求参数 页面。其中，此接口的 Action 字段为 CreateDsaHost。

参数名称	必选	类型	描述
host	是	String	待新增的加速域名 host。
origin	是	String	回源地址，支持配置一个域名，或配置多个源站 IP（支持 '域名：port' 和 'ip:port' 类型），端口仅支持 >0 且 ⩽ 65535，多 IP 时采用逗号分隔。
projectId	否	Unsigned	项目 ID，指域名要添加入的项目的对应 ID，查看项目 ID 。
https	否	String	HTTPS 配置（JSON），默认不启用 HTTPS，配置格式见下文 **HTTPS 配置格式**。
rspHeader	否	String	自定义返回头部（JSON），配置格式见下文 **HTTPS 配置格式**。
strategyWeight	否	String	权重回源配置。源站仅允许配置 IP，且 key 值与 “origin” 字段配置源站相同，value 值为权重，权重范围1 – 100。示例：若 origin = "1.1.1.1:8080,2.2.2.2"，则权重 = {"1.1.1.1:8080":10,"2.2.2.2":20}。
backupOrigin	否	String	备份源站配置。仅支持一主一备，且备份源站仅允许配置 IP。示例：["1.2.3.4:8080"]。

HTTPS 配置格式

参数名称	是否必选	类型	描述
type	是	Int	HTTPS 开关： 0：表示关闭。 2：表示启用，并且采用 HTTP 回源。 4：表示启用，并采用协议跟随回源，默认关闭 HTTPS 配置。
crt	否	String	当启用 HTTPS，并且采用自有证书时，必须配置证书信息。
private_key	否	String	当启用 HTTPS，并且采用自有证书时，必须配置私钥信息。

cert_id	否	String	当启用 HTTPS，并且采用腾讯云托管证书时，必须配置证书 ID 信息。
message	否	String	备注信息。

> **Note:**
> 当启用 HTTPS 配置，且采用自有证书时，由于需要上传证书信息，请**采用 POST 方式提交**。

自定义头部配置格式

自定义回源头部采用 JSON 格式配置，配置格式如下：

```
{
    "key_name_1":"value_1",
    "key_name_2":"value_2",
    ...
    "key_name_N":"value_N"
}
```

> **Note:**
> 当设置自定义头部时，请尽量**采用 POST 方式提交**，防止 URL 长度越界问题。

输出参数

参数名称	类型	描述
code	Int	公共错误码 0：表示成功 其他值：表示失败 详情请参见错误码页面的 公共错误码 。
message	String	模块错误信息描述，与接口相关。
codeDesc	String	英文错误信息，或业务侧错误码。
data	Object	输出结果，若域名添加成功，则返回域名配置信息。

data 字段说明

参数名称	类型	描述
host_id	Int	域名接入 ECDN 后的唯一标识 ID。
app_id	Int	域名归属人的 AppID。
project_id	Unsigned	域名所属的项目 ID。
host	String	加速域名。
cname	String	域名接入 ECDN 的 CNAME 别名。
status	String	域名当前状态 online：启用 offline：关闭。
progress	Int	配置部署状态 0：表示部署完毕 1：表示正在部署中。
mode	Int	域名封禁状态，若为 '0'，则表示域名未被封禁。
locked	Int	运维锁状态 0：表示未锁定，用户可直接修改配置 1：表示锁定状态，需要提交工单审核后才能修改配置。

deleted	Int	域名是否删除 0：表示域名未删除 1：表示域名已删除。
origin	String	源站地址。
fwd_host	String	回源 host 配置。
rsp_header	Object	自定义返回头部配置信息。
https	Object	HTTPS 配置信息，详情见下文 **HTTPS 字段说明**。
message	String	信息描述。
create_time	String	域名创建时间。
update_time	String	域名上一次配置更新时间。
strategy_weight	String	回源权重。
backup_origin	String	备份源站。

HTTPS 字段说明

参数名称	类型	描述
type	Unsigned	HTTPS 开关 0：表示关闭 2：表示启用，并且采用 HTTP 回源 4：表示启用，并采用协议跟随回源。
cert_id	String	腾讯云托管证书 ID，当使用腾讯云托管证书时返回此参数。
cn	String	证书域名。
expire	String	SSL 证书过期时间。
message	String	证书备注信息。
http2	String	HTTP 2.0 开关 on：表示开启 off：表示关闭。

> **Note:**
> 未在上述文档中说明的字段均为**无效字段**，可直接忽略。

代码示例

配置需求示例

```
host: dsa.qcloud.com
projectId: 0
origin: origin.dsa.qcloud.com: 8080
```

GET 请求

GET 请求需要将所有参数都加在 URL 后：

```
https://dsa.api.qcloud.com/v2/index.php?
Action=CreateDsaHost
&SecretId=XXXXXXXXXXXXXXXXXXXXXXXX
&Timestamp=1462440051
&Nonce=123456789
```

```
&Signature=XXXXXXXXXXXXXXXXXXXXXXXXXXX
&host=dsa.qcloud.com
&projectId=0
&origin=origin.dsa.qcloud.com:8080
```

POST 请求

POST 请求时，参数填充在 HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array(
    'Action' => 'CreateDsaHost',
      'SecretId' => 'SecretId',
      'Timestamp' => 1462782282,
      'Nonce' => 123456789,
     'Signature' => 'Signature',
    'host' => 'host',
    'origin' => 'origin',
    'projectId' => 'project_id',
  );
```

返回结果示例

创建成功

```
{
    "code": 0,
    "message": "",
    "codeDesc": "Success",
    "data": {
        "host_id":3827,
        "app_id":1251396975,
        "project_id":0,
        "host":"arnoapi2.elliotxing.com",
        "cname":"arnoapi2.elliotxing.com.dsa.dnsv1.com",
        "status":"online",
        "progress":1,
        "mode":0,
        "locked":0,
        "deleted":0,
        "origin":"8.8.8.8:8080",
        "fwd_host":"arnoapi2.elliotxing.com",
        "rsp_header":[],
        "https":{"type":0},
        "message":"",
        "create_time":"2018-05-27 19:42:04",
        "update_time":"2018-05-27 19:42:06"
        }
}
```

创建失败

```
{
    "code": 4000,
    "message": "(2001)您提交的域名已经存在，请勿重复添加",
    "codeDesc": "InvalidParameter"
}
```

启动DSA域名

最近更新时间：2020-03-13 18:01:04

接口描述

本接口（ OnlineDsaHost ）用于启动指定的 ECDN 加速域名接口请求域名：dsa.api.qcloud.com

> 1）使用此接口，一次仅支持启动一个域名；2）该功能主要针对处于关闭状态的域名；3）该接口调用频次上限为100次 / 分钟，超出上限则会返回错误，请勿高频调用。

输入参数

以下请求参数列表仅列出了接口请求参数，正式调用时需要加上 公共请求参数 ，见公共请求参数页面。其中，此接口的 Action 字段为 OnlineDsaHost。

参数名称	必选	类型	描述
hostId	是	Unsigned	域名接入 ECDN 后的标识 ID,可通过 查询域名列表信息 接口获取

输出参数

参数名称	类型	描述
code	Int	公共错误码，0 表示成功，其他值表示失败。详见错误码页面的 公共错误码 。
message	String	模块错误信息描述，与接口相关
codeDesc	String	英文错误信息，或业务侧错误码。

示例

1. 配置需求示例

启用 dsa.qcloud.com 域名的 ECDN 加速服务：

> host：dsa.qcloud.comhostId：1234

2. GET 请求

GET请求需要将所有参数都加在URL后：

```
https://dsa.api.qcloud.com/v2/index.php?
Action=OnlineDsaHost
&SecretId=XXXXXXXXXXXXXXXXXXXXXXXXXX
&Timestamp=1462440051
&Nonce=123456789
&Signature=XXXXXXXXXXXXXXXXXXXXXXXXXX
&hostId=1234
```

3. POST 请求

POST 请求时，参数填充在 HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array (
  'Action' => 'OnlineDsaHost',
  'SecretId' => 'SecretId',
  'Timestamp' => 1462782282,
  'Nonce' => 123456789,
  'Signature' => 'Signature',
  'hostId' => 1234
)
```

4. 返回结果示例

启用成功

```
{
    "code": 0,
    "message": "",
    "codeDesc": "Success"
}
```

启用失败

```
{
    "code": 4000,
    "message": "(2000)加速域名不存在",
    "codeDesc": "InvalidParameter"
}
```

关闭DSA域名

最近更新时间：2020-03-13 18:01:39

接口描述

本接口（ OfflineDsaHost ）用于关闭指定域名的 ECDN 加速服务。接口请求域名：dsa.api.qcloud.com

1）使用该接口，一次仅允许关闭一个域名；2）域名关闭后，Cname 域名解析会改回源站，HOST 访问到 CDN 节点的请求会统一返回 404；
3）该接口调用频次上限为100次 / 分钟，超出上限则会返回错误，请勿高频调用。

输入参数

以下请求参数列表仅列出了接口请求参数，正式调用时需要加上 公共请求参数 ，见公共请求参数页面。其中，此接口的 Action 字段为 OfflineDsaHost。

参数名称	必选	类型	描述
hostId	是	Unsigned	域名接入 ECDN 后的标识 ID,可通过 查询域名列表信息 接口获取

输出参数

参数名称	类型	描述
code	Int	公共错误码，0 表示成功，其他值表示失败。详见错误码页面的 公共错误码 。
message	String	模块错误信息描述，与接口相关
codeDesc	String	英文错误信息，或业务侧错误码。

示例

1. 配置需求示例

关闭 dsa.qcloud.com 域名的 ECDN 加速服务：

host：dsa.qcloud.comhostId：1234

2. GET 请求

GET请求需要将所有参数都加在URL后：

```
https://dsa.api.qcloud.com/v2/index.php?
Action=OfflineDsaHost
&SecretId=XXXXXXXXXXXXXXXXXXXXXXXX
&Timestamp=1462440051
&Nonce=123456789
&Signature=XXXXXXXXXXXXXXXXXXXXXXXX
&hostId=1234
```

3. POST 请求

POST 请求时，参数填充在 HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array (
  'Action' => 'OfflineDsaHost',
  'SecretId' => 'SecretId',
  'Timestamp' => 1462782282,
  'Nonce' => 123456789,
  'Signature' => 'Signature',
  'hostId' => 1234
)
```

4. 返回结果示例

关闭成功

```
{
    "code": 0,
    "message": "",
    "codeDesc": "Success"
}
```

关闭失败

```
{
    "code": 4000,
    "message": "请求失败，参数[hostId]不能为空。",
    "codeDesc": "InvalidParameter"
}
```

删除加速域名

最近更新时间：2020-03-13 18:02:27

接口描述

本接口（ DeleteDsaHost ）用于删除指定域名的 ECDN 加速服务。接口请求域名： `dsa.api.qcloud.com`

> **Note:**
> 1. 使用该接口，一次仅允许关闭一个域名。
> 2. 只有处于关闭状态的域名才允许进行域名删除操作。
> 3. 该接口调用频次上限为100次 / 分钟，超出上限则会返回错误，请勿高频调用。

输入参数

以下请求参数列表仅列出了接口请求参数，正式调用时需要加上 公共请求参数 ，见公共请求参数页面。其中，此接口的 Action 字段为 DeleteDsaHost。

参数名称	必选	类型	描述
hostId	是	Unsigned	域名接入 ECDN 后的标识 ID，可通过 查询域名列表信息 接口获取

输出参数

参数名称	类型	描述
code	Int	公共错误码，0表示成功，其他值表示失败。详见错误码页面的 公共错误码 。
message	String	模块错误信息描述，与接口相关
codeDesc	String	英文错误信息，或业务侧错误码。

示例

1. 配置需求示例

删除 dsa.qcloud.com 域名的 ECDN 加速服务：

```
host：dsa.qcloud.comhostId：1234
```

2. GET 请求

GET 请求需要将所有参数都加在 URL 后：

```
https://dsa.api.qcloud.com/v2/index.php?
Action=DeleteDsaHost
&SecretId=XXXXXXXXXXXXXXXXXXXXXXXX
&Timestamp=1462440051
&Nonce=123456789
&Signature=XXXXXXXXXXXXXXXXXXXXXXXX
&hostId=1234
```

2. POST 请求

POST 请求时，参数填充在 HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array (
  'Action' => 'DeleteDsaHost',
  'SecretId' => 'SecretId',
  'Timestamp' => 1462782282,
  'Nonce' => 123456789,
  'Signature' => 'Signature',
  'hostId' => 1234
)
```

4. 返回结果示例

删除成功

```
{
    "code": 0,
    "message": "",
    "codeDesc": "Success"
}
```

删除失败

```
{
    "code": 4000,
    "message": "(2005)请求的域名已下线",
    "codeDesc": "InvalidParameter"
}
```

修改域名配置

最近更新时间：2021-01-07 14:22:52

接口描述

本接口（UpdateDsaHostInfo）用于修改域名对应源站配置。

接口请求域名：`dsa.api.qcloud.com`

- 状态为下线、部署中的域名不可修改配置。
- 一次仅支持修改一个域名对应的配置。
- 一次可修改同一个域名的多个配置项。
- 该接口调用频次上限为100次/分钟，超出上限则会返回错误，请勿高频调用。

可支持配置

- 修改源站配置。
- 修改域名所属项目。
- 修改 HTTPS 配置。
- 修改域名缓存规则。
- 设置回源host。
- 设置加速区域。
- 设置 response header。

输入参数

以下请求参数列表仅列出了接口请求参数，正式调用时需要加上公共请求参数，详情请参见 公共请求参数 页面。其中，此接口的 Action 字段为 UpdateDsaHostInfo。

参数名称	是否必选	类型	描述
hostId	是	Int	域名接入 ECDN 后的标识 ID。</br>您可通过 查询域名列表信息 接口查询域名的 hostId。
projectId	否	Unsigned	项目 ID，指域名要添加入的项目的对应 ID，查看项目 ID 。
origin	否	String	源站地址，源站配置可填写一个源站域名或者多个源站 IP。 可支持 domain:port 或者 ip:port 类型配置。 端口可配置区间0 - 65535，不设置端口时默认 HTTP 回80，HTTPS 回443，设置多个源站时仅允许设置一个相同端口。 示例：`1.1.1.1,2.2.2.2` 或者 `1.1.1.1:8080,2.2.2.2:8080` 。
https	否	String	HTTPS 配置信息（JSON），需要修改 HTTPS 配置时使用，查看 HTTPS 配置 。
cache	否	String	缓存规则配置（json），查看缓存配置 。
area	否	String	加速区域：默认只有中国大陆加速， ECDN 开启中国境外加速服务，立即申请使用资格 >> 开通后可以配置其他区域： mainland：中国大陆 oversea：中国港澳台及海外地区 global：全球
furlcache	否	String	开启完整URL缓存。 on 开启，表示不过滤问号后参数缓存。 off 关闭，表示过滤问号后参数缓存。
rspHeader	否	String	自定义头部信息，需要修改自定义回源头部时使用，查看自定义头部配置 。

strategyWeight	否	String	权重回源配置。配置格式为 `{"key1":value,"key2":value}` 。 源站仅允许配置 IP，且 key 值与“origin”字段配置源站相同，value 值为权重，权重范围1 – 100。 示例：若origin = `"1.1.1.1:8080,2.2.2.2:8080"` ，则权重 = `{"1.1.1.1:8080":10,"2.2.2.2:8080":20}` 。
backupOrigin	否	String	备份源站配置，仅允许在 origin 配置 IP 下可以配置，仅支持配置 IP。 可支持 domain:port 或者 ip:port 类型配置。 端口可配置区间0 - 65535，不设置端口时默认 HTTP 走80，HTTPS 回443，设置多个源站时仅允许设置一个相同端口。 示例： `1.1.1.1,2.2.2.2` 或者 `1.1.1.1:8080,2.2.2.2:8080` 。
fwdHost	否	String	回源 Host 配置。需要修改回源 Host 时使用。示例： `fwdHost = www.abc.com` 。

HTTPS 配置格式

参数名称	是否必选	类型	描述
type	是	Int	HTTPS 开关： 0：表示关闭。 2：表示启用，并且采用 HTTP 回源。 4：表示启用，并采用协议跟随回源。
crt	否	String	PEM 格式证书,当启用 HTTPS 时必须配置证书。
private_key	否	String	PEM 格式密钥，当启用 HTTPS 时必须配置私钥。
cert_id	否	String	启用 HTTPS 后，需要采用腾讯云托管证书时，必须配置证书 ID 信息，证书托管配置只需传入此参数即可。
message	否	String	备注信息。

> **Note:**
> 当需要提交证书信息时，请**采用 POST 方式提交**，防止出现 URL 长度越界问题。

缓存配置格式

参数名称	是否必选	类型	描述
type	是	Int	0：全部文件缓存配置 1：按文件后缀类型缓存配置 2：文件夹/目录路径缓存配置 3：具体 URL 缓存配置，例如 `folder/*.jpg` 类型的正则匹配 5：首页缓存设置
rule	是	String	具体类型的缓存规则，如 `.jpg;.png` 多个文件以 `;` 隔开 (type == 0时只能配置为 all)
time	是	Int	缓存时间数值设置，其中0表示不缓存。与 unit 参数共同设置缓存时间
unit	是	String	时间单位：d天，h时，m分，s秒。与 time 参数共同设置缓存时间

配置示例

缓存规则优先级，从上往下优先级提高，最后一条优先级最高。

```
[{"type":0,"rule":"all","time":0,"unit":"d"},
{"type":1,"rule":".txt","time":1,"unit":"d"},
{"type":2,"rule":"/test;/abc","time":1,"unit":"h"},
{"type":3,"rule":"/test/*.jpg","time":1,"unit":"m"},
```

```
{"type":5,"rule":"/","time":1,"unit":"s"}]
```

> **Note:**
> 当需要 Cache 配置信息时，建议**采用 POST 方式提交**，防止URL 长度越界。

自定义头部配置格式

自定义回源头部采用 JOSN 格式配置，配置格式如下：

```
{
    "key_name_1":"value_1",
    "key_name_2":"value_2",
    ...
    "key_name_N":"value_N"
}
```

> **Note:**
> 当设置自定义头部时，请**采用 POST 方式提交**，防止 URL 长度越界。

输出参数

参数名称	类型	描述
code	Int	公共错误码 0：表示成功。 其他值：表示失败。 详情请见错误码页面的 公共错误码 。
message	String	模块错误信息描述，与接口相关。
codeDesc	String	英文错误信息，或业务侧错误码。
data	Object	输出结果，返回域名修改后的配置信息。

data 字段说明

参数名称	类型	描述
host_id	Int	域名接入 ECDN 后的唯一标识 ID。
app_id	Int	域名归属人的 AppID。
project_id	Unsigned	域名所属的项目 ID。
host	String	加速域名。
cname	String	域名接入 ECDN 的 CNAME 别名。
status	String	域名当前状态 online：启用。 offline：关闭。
progress	Int	配置部署状态 0：表示部署完毕。 1：表示正在部署中。
mode	Int	域名封禁状态，若为 '0'，则表示域名未被封禁。
locked	Int	运维锁状态 0：表示未锁定，用户可直接修改配置。 1：表示锁定状态，需要提交工单审核后才能修改配置。
deleted	Int	域名是否删除 0：表示域名未删除。 1：表示域名已删除。
origin	String	源站地址。

fwd_host	String	回源 host 配置。
rsp_header	Array	自定义返回头部配置信息。
https	Object	HTTPS 配置信息，详情请参见下文 HTTPS 字段说明 。
cache	String	缓存配置，查看缓存配置 。
area	String	加速区域。
furlcache	String	是否开启完整 URL 缓存。
message	String	信息描述。
create_time	String	域名创建时间。
update_time	String	域名上一次配置更新时间。
strategy_weight	String	回源权重。
backup_origin	String	备份源站。

HTTPS 字段说明

参数名称	类型	描述
type	Unsigned	HTTPS 开关： 0：表示关闭。 2：表示启用，并且采用 HTTP 回源。 4：表示启用，并采用协议跟随回源。
cert_id	String	腾讯云托管证书 ID，当使用腾讯云托管证书时返回此参数。
cn	String	证书域名。
expire	String	SSL 证书过期时间。
message	String	证书备注信息。
http2	String	HTTP 2.0 开关： on：表示开启。 off：表示关闭。

> **Note:**
> 未在上述文档中说明的字段均为**无效字段**，可直接忽略。

配置示例

将 `dsa.qcloud.com` **的源站地址修改为** `source2.dsa.qcloud.com` **。**
启用 dsa.qcloud.com 域名的 HTTPS 配置，采用自有证书和协议跟随方式回源。
启用缓存，.txt 类型缓存1天，/test 和 /abc 目录缓存1小时，/test/ 目录下所有 jpg 文件缓存1分钟，首页缓存1秒，其他文件不缓存。

```
host: dsa.qcloud.com
hostId: 1234
origin: source2.dsa.qcloud.com
回源方式：协议跟随
证书信息：  ***************
证书私钥：  ***************
```

GET 请求

GET 请求需要将所有参数都加在 URL 后：

当需要上传证书信息时和配置缓存规则时，由于URL长度限制，建议不要采用 Get 方式提交配置修改请求。

```
https://dsa.api.qcloud.com/v2/index.php?
Action=UpdateDsaHostInfo
&SecretId=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
&Timestamp=1462520137
&Nonce=123456789
&Signature=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
&hostId=1234
&origin=source2.dsa.qcloud.com
&cache=[{"type":0,"rule":"all","time":0,"unit":"d"},{"type":1,"rule":".txt","time":1,"unit":"d"},
{"type":2,"rule":"/test;/abc","time":1,"unit":"h"},
{"type":3,"rule":"/test/*.jpg","time":1,"unit":"m"},{"type":5,"rule":"/","time":1,"unit":"s"}]
```

POST 请求

POST 请求时，参数填充在 HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array (
  'Action' => 'UpdateDsaHostInfo',
  'SecretId' => 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX',
  'Timestamp' => 1462782282,
  'Nonce' => 123456789,
  'Signature' => 'XXXXXXXXXXXXXXXXXXXXXXXX',
  'hostId' => '1234',
  'origin' => 'source2.dsa.qcloud.com',
  'https' => '{"type":4,"crt":"证书内容","private_key":"证书私钥内容"}',
  'cache' => ' [{"type":0,"rule":"all","time":0,"unit":"d"},
{"type":1,"rule":".txt","time":1,"unit":"d"},{"type":2,"rule":"/test;/abc","time":1,"unit":"h"},
{"type":3,"rule":"/test/*.jpg","time":1,"unit":"m"},{"type":5,"rule":"/","time":1,"unit":"s"}]'
)
```

返回结果示例

修改成功

```
{
    "code": 0,
    "message": "",
    "codeDesc": "Success",
    "data": {
        "host_id": 1234,
        "app_id": "*******",
        "project_id": 0,
        "host": "dsa.qcloud.com",
        "cname": "dsa.qcloud.com.dsa.dnsv1.com",
        "status": "online",
        "progress": 0,
        "mode": 0,
```

```
        "locked": 0,
        "deleted": 0,
        "origin": "source2.dsa.qcloud.com",
        "fwd_host": "dsa.qcloud.com",
        "rsp_header": [],
        "https": {"type": 0},
        "ip_access":{"type":"off","ips":[]},
        "ip_freq_limit":0,
        "message": "",
        "create_time": "2017-11-17 11:07:08",
        "update_time": "2018-01-28 12:52:02"
}
{
    "code": 0,
    "message": "",
    "codeDesc": "Success",
    "data": {
              "host_id": 1234,
            "app_id": "*******",
            "project_id": 0,
            "host": "dsa.qcloud.com",
            "cname": "dsa.qcloud.com.dsa.dnsv1.com",
        "status": "online",
        "progress": 0,
        "https_progress": 0,
        "mode": 0,
        "locked": 0,
        "deleted": 0,
            "origin": "source2.dsa.qcloud.com",
        "fwd_host": "dsa.qcloud.com",
                "rsp_header": [],
            "https": {
                "type": 4,
                "crt":"***************",
                "private_key":"***************"
           },
        },
        "ip_access": {
            "ips": [],
            "type": "off"
        },
        "ip_freq_limit": 0,
        "strategy_weight": [],
        "backup_origin": [],
        "message": "",
        "create_time": "2019-07-09 15:04:41",
        "update_time": "2019-09-20 17:52:07",
        "area": "global",
        "cache": [
            {
                "type": 0,
                "rule": "all",
                "time": 0,
                "unit": "d"
            },
            {
                "type": 1,
```

```
                "rule": ".txt",
                "time": 1,
                "unit": "d"
            },
            {
                "type": 2,
                "rule": "/test;/abc",
                "time": 1,
                "unit": "h"
            },
            {
                "type": 3,
                "rule": "/test/*.jpg",
                "time": 1,
                "unit": "m"
            },
            {
                "type": 5,
                "rule": "/",
                "time": 1,
                "unit": "s"
            }
        ],
        "icp_status": 1,
        "furl_cache": "off"
    }
}
}
```

数据查询相关接口
数据查询接口概览

最近更新时间：2020-05-29 11:35:21

ECDN 数据查询接口是用于查询动态网络加速服务用量信息，支持以下查询功能：

接口名称	功能描述
GetDsaStatistics	监控明细数据查询接口
GetDsaHostStatistics	统计数据查询接口

接口说明

1、支持带宽、流量、请求次数、访问延时、状态码等多个指标的监控数据和统计数据查询；
2、监控数据是指根据查询时间段和查询时间粒度，返回详细的数据分布情况，如查询5分钟粒度的状态码分布情况；
3、统计数据是指根据查询时间段，返回该时间段内的统计数据，如查询一个月内的总请求次数。

监控数据查询示例

示例名称	示例描述
流量监控数据查询	查询指定时间区间的实时流量分布数据
带宽监控数据查询	查询指定时间区间的实时带宽分布数据

监控数据查询

监控数据查询

最近更新时间：2021-01-22 10:57:21

接口描述

本接口用于查询指定时间区间的监控数据，可用于观察域名访问变化情况。接口请求域名：`dsa.api.qcloud.com` 接口名：`GetDsaStatistics`

注意：

1. 该接口调用频次上限为100次/分钟，请勿高频调用。
2. 可一次查询多个项目或多个域名的监控明细数据。
3. 可一次查询多个监控指标的明细数据。
4. 支持最长查询时间跨度90天。

支持查询的监控指标说明

统计指标	指标名称	指标说明
访问流量	flux	查询域名访问流量监控数据，单位：Byte
访问带宽	bandwidth	查询域名访问带宽监控数据，单位：bps
请求次数	request	查询域名访问次数监控数据，单位：次数
访问延时	delay	查询域名平均访问延时，单位：ms
状态码	200、400、500等	查询域名状态码监控数据，单位：次数

输入参数

以下请求参数列表仅列出了接口请求参数，正式调用时需要加上公共请求参数，详情请参见 公共请求参数 页面，此接口的 Action 字段为 `GetDsaStatistics` 。

参数	必选	类型	取值示例	描述
metrics	是	String	["flux","request"]	统计指标，可一次提交多个查询指标，按 JSON 格式提交
projects	否	Unsigned	[1001853]	项目ID列表，查看项目ID 按 JSON 格式提交
hosts	否	String	["test.qcloud.com"]	域名列表，当根据域名查询数据时使用，按 JSON 格式提交
startDate	否	String	2018-04-19	开始时间，默认为查询当日日期 格式：YY-MM-DD
endDate	否	String	2018-04-20	结束时间，默认为查询当日日期 格式：YY-MM-DD
granularity	否	Unsigned	15	查询时间粒度（分钟），详情请参见查询时间粒度规则

注意：

1. 未指定查询目标时，默认查询账号下全部域名的合并统计数据。
2. 当查询多个项目或多个域名时，为了防止查询参数过长导致 URL 越界，建议尽量采用 POST 方式提交请求。

查询时间粒度规则

起止时间跨度	默认粒度	支持查询时间粒度
1 天	1	1，5，15，30，60，120，240，1440
2 ~ 3 天	15	15，30，60，120，240，1440
4 ~ 7 天	30	30，60，120，240，1440
8 ~ 90 天	60	60，120，240，1440

数据汇总说明：

1. 监控数据每1分钟为一个基本数据采样点。
2. 访问流量、访问次数和状态码监控数据按照时间累加合并。
3. 访问延时按照平均数合并。
4. 时间粒度为5分钟的访问带宽，以1分钟粒度带宽为样本值，取5分钟内的平均带宽值。
5. 时间粒度大于5分钟的访问带宽，以5分钟粒度带宽为样本值，取最大带宽值。

输出参数

参数	类型	描述
code	Int	公共错误码 0：表示成功 其他值：表示失败 详情请参见错误码页面的 公共错误码
message	String	模块错误信息描述，与接口相关
codeDesc	String	英文错误信息，或业务侧错误码
data	Array	详细查询数据信息，详情请参见 data 字段说明

⚠ 注意：

1. **当未指定查询目标时，默认按照账号维度返回全部域名合并的监控明细数据。**
2. **当指定查询域名或查询项目时，按域名维度返回每个域名的监控明细数据。**

data 字段说明

参数	数值类型	描述
datetime	String	数据时间刻度，如：2018-04-19 01:00:00
bandwidth	Unsigned	查询指标包含带宽时返回该字段，数据单位：bps
flux	Unsigned	查询指标包含流量时返回该字段，数据单位：byte
request	Unsigned	查询指标包含请求次数时返回该字段，数据单位：次数
delay	Unsigned	查询指标包含访问延时时返回该字段，数据单位：ms
状态码	Unsigned	查询指定状态码的响应次数，数据单位：次数

示例代码

查询账号带宽监控数据

需求说明

查询账号下全部域名2018-04-19的带宽监控数据,查询时间粒度60分钟。

GET 请求

GET 请求需要将所有参数都加在 URL 后：

```
https://dsa.api.qcloud.com/v2/index.php?
Action=GetDsaStatistics
&SecretId=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
&Timestamp=1524279600
&Nonce=123456789
&Signature=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
&metrics=["bandwidth"]
&startDate=2018-04-19
&endDate=2018-04-19
&granularity=60
```

> **注意：**
> 为了防止查询参数过长，导致 URL 越界，该接口默认采用 POST 方式提交查询请求。

POST 请求

POST 请求时，参数填充在 HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array (
  'Action' => 'GetDsaHostLogs',
  'SecretId' => 'SecretId',
  'Timestamp' => 1524279600,
  'Nonce' => 123456789,
  'Signature' => 'Signature',
  'metrics' => '["bandwidth"]',
  'startDate' => 2018-04-19,
  'endDate' => 2018-04-19,
  'granularity' => 60
)
```

返回结果示例

查询成功

```
{
    "code": 0,
    "message": "",
    "codeDesc": "Success",
    "data": [
        {
            "datetime": "2018-04-19 00:00:00",
            "bandwidth": 589746515
```

```
        },
        {
            "datetime": "2018-04-19 01:00:00",
            "bandwidth": 489746515
        },
        {
            "datetime": "2018-04-19 02:00:00",
            "bandwidth": 375489625
        },
         ......
        {
            "datetime": "2018-04-19 23:00:00",
            "bandwidth": 589746515
        }
    ]
}
```

查询失败

```
{
    "code": 4100,
    "message": "鉴权失败，请参考文档中鉴权部分。",
    "codeDesc": "AuthFailure"
}
```

查询指定域名的请求次数和访问流量监控数据

需求说明

查询域名 `test1.dsa.qcloud.com` 和 `test2.dsa.qcloud.com` 在2018-04-19的请求数和流量监控数据，查询时间粒度5分钟。

GET 请求

GET 请求需要将所有参数都加在 URL 后：

```
https://dsa.api.qcloud.com/v2/index.php?
Action=GetDsaStatistics
&SecretId=XXXXXXXXXXXXXXXXXXXXXXXXXXXX
&Timestamp=1524279600
&Nonce=123456789
&Signature=XXXXXXXXXXXXXXXXXXXXXXXXXXXX
&metrics=["request","flux"]
&hosts=["test1.dsa.qcloud.com","test2.dsa.qcloud.com"]
&startDate=2018-04-19
&endDate=2018-04-19
&granularity=5
```

> **注意：**
> 当查询域名较多时，为了防止查询参数过长，导致 URL 越界，该接口建议采用 POST 方式提交查询请求。

POST 请求

POST 请求时，参数填充在 HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array (
  'Action' => 'GetDsaHostLogs',
  'SecretId' => 'SecretId',
  'Timestamp' => 1524279600,
  'Nonce' => 123456789,
  'Signature' => 'Signature',
  'metrics' => '["request","flux"]',
  'hosts' => ["test1.dsa.qcloud.com","test2.dsa.qcloud.com"],
  'startDate' => 2018-04-19,
  'endDate' => 2018-04-19,
  'granularity' => 5
)
```

返回结果示例

查询成功

```
{
    "code": 0,
    "message": "",
    "codeDesc": "Success",
    "data": {
        "test1.dsa.qcloud.com":[
            {
                "datetime": "2018-04-19 00:00:00",
                "request": 589746515,
                "flux": 231231231
            },
            {
                "datetime": "2018-04-19 00:05:00",
                "request": 489746515,
                "flux": 524864864
            },
            {
                "datetime": "2018-04-19 00:10:00",
                "request": 375489625,
                "flux": 33545687
            },
            ......
            {
                "datetime": "2018-04-19 23:55:00",
                "request": 589746515,
                "flux": 52897464
            }
        ],
        "test2.dsa.qcloud.com":[
            {
                "datetime": "2018-04-19 00:00:00",
                "request": 589746515,
                "flux": 25631567
            },
            {
                "datetime": "2018-04-19 00:05:00",
                "request": 489746515,
                "flux": 45313876
```

```
            },
            {
                "datetime": "2018-04-19 00:10:00",
                "request": 375489625,
                "flux": 5789456314
            },
            ......
            {
                "datetime": "2018-04-19 23:55:00",
                "request": 589746515,
                "flux": 150789457
            }
        ]
    }
}
```

查询失败

```
{
    "code": 4100,
    "message": "鉴权失败，请参考文档中鉴权部分。",
    "codeDesc": "AuthFailure"
}
```

流量监控数据查询

最近更新时间：2019-12-31 14:52:22

接口描述

本接口示例用于指导流量监控数据查询。
域名：`dsa.api.qcloud.com`
接口名：`GetDsaStatistics`

> **注意：**
> 1. 该接口调用频次上限为100次/分钟，请勿高频调用。
> 2. 可一次查询多个项目或多个域名的实时合并统计数据。
> 3. 每1分钟为一个统计点，流量数据按照流量值累加统计。
> 4. 最多支持查询最近60天的数据。

输入参数

以下请求参数列表仅列出了接口请求参数，正式调用时需要加上公共请求参数，详情请参见 公共请求参数 页面，此接口的 Action 字段为 `GetDsaStatistics` 。

参数	必选	类型	示例	描述
metrics	是	String	["flux"]	统计指标 按 JSON 格式提交
projects	否	Unsigned	[1001853]	项目ID列表，查看项目ID 按 JSON 格式提交
hosts	否	String	test.qcloud.com	域名列表 按 JSON 格式提交
startDate	否	String	2018-04-19	开始时间，默认为查询当日日期 格式：YY-MM-DD
endDate	否	String	2018-04-20	结束时间，默认为查询当日日期 格式：YY-MM-DD
granularity	否	Unsigned	15	查询时间粒度（分钟）

> **注意：**
> 1. 未指定查询目标时，默认按账号维度汇总流量数据。
> 2. 当指定查询项目列表时，按项目维度汇总流量数据。
> 3. 当指定查询域名列表时，按域名维度汇总流量数据。
> 4. 域名列表优先级高于项目列表，当同时指定查询项目列表和域名列表时，按指定的域名列表响应。
> 5. 当查询多个项目或多个域名时，为了防止查询参数过长导致 URL 越界，建议采用 POST 方式提交请求。

查询时间粒度规则

起止时间跨度	默认粒度	支持查询时间粒度
1 天	1	1，5，15，30，60，120，240，1440
2 ~ 3 天	15	15，30，60，120，240，1440

4～7天	30	30，60，120，240，1440
8～90天	60	60，120，240，1440

说明：
流量数据每1分钟为一个样本点，按照流量值累加统计。

响应参数

参数	类型	描述
code	Int	公共错误码 0：表示成功 其他值：表示失败 详情请参见 错误码 文档中的公共错误码说明
message	String	模块错误信息描述，与接口相关
codeDesc	String	英文错误信息，或业务侧错误码
data	Array或Object	详细查询数据信息，详细请参见 data 字段说明

data 字段说明

1、按账号维度查询时，data 内容是一个数组，每个数组元素包含一个监控点的流量明细数据，数组字段如下：

参数	类型	描述
datetime	String	数据时间刻度 例如：2018-04-19 01:00:00
flux	Unsigned	流量明细值，单位：byte

2、按项目维度查询时，data 内容是一个对象，每个对象元素是一个项目的流量明细数组，具体信息如下：

参数	类型	描述
项目ID	Array	项目详细流量监控数据数组，数组字段如下： “datetime”：数据时间刻度 “flux”：流量明细值，单位：byte

3、按域名维度查询时，data 内容是一个对象，每个对象元素是一个域名的流量明细数组，具体信息如下：

参数	类型	描述
域名	Array	域名详细流量监控数据数组，数组字段如下： “datetime”：数据时间刻度 “flux”：流量明细值，单位：byte

示例一 按账号维度查询

查询需求说明

查询账号下全部域名在2018-04-19的流量监控数据，查询时间粒度60分钟。

GET 请求

GET 请求需要将所有参数都加在 URL 后：

```
https://dsa.api.qcloud.com/v2/index.php?
Action=GetDsaStatistics
&SecretId=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
&Timestamp=1524279600
&Nonce=123456789
&Signature=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
&metrics=["flux"]
&startDate=2018-04-19
&endDate=2018-04-19
&granularity=60
```

POST 请求示例

POST 请求时，参数填充在 HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array (
  'Action' => 'GetDsaHostLogs',
  'SecretId' => 'SecretId',
  'Timestamp' => 1524279600,
  'Nonce' => 123456789,
  'Signature' => 'Signature',
  'metrics' => '["flux"]',
  'startDate' => 2018-04-19,
  'endDate' => 2018-04-19,
  'granularity' => 60
)
```

返回结果示例

查询成功

```
{
    "code": 0,
    "message": "",
    "codeDesc": "Success",
    "data": [
        {
            "datetime": "2018-04-19 00:00:00",
            "flux": 874271534
        },
        {
            "datetime": "2018-04-19 01:00:00",
            "flux": 744647032
        },
        {
            "datetime": "2018-04-19 02:00:00",
            "flux": 648965244
        },
         ......
        {
            "datetime": "2018-04-19 23:00:00",
```

```
            "flux": 974651168
        }
    ]
}
```

查询失败

```
{
    "code": 4100,
    "message": "鉴权失败，请参考文档中鉴权部分。",
    "codeDesc": "AuthFailure"
}
```

示例二 按项目维度查询

查询需求说明

同时查询两个项目在2018-04-19的总使用流量，项目ID分别为100873和1006363。

GET 请求

GET 请求需要将所有参数都加在 URL 后：

```
https://dsa.api.qcloud.com/v2/index.php?
Action=GetDsaStatistics
&SecretId=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
&Timestamp=1524279600
&Nonce=123456789
&Signature=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
&metrics=["flux"]
&projects=[100873,1006363]
&startDate=2018-04-19
&endDate=2018-04-19
&granularity=1440
```

POST 请求示例

POST 请求时，参数填充在 HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array (
  'Action' => 'GetDsaHostLogs',
  'SecretId' => 'SecretId',
  'Timestamp' => 1524279600,
  'Nonce' => 123456789,
  'Signature' => 'Signature',
  'metrics' => '["flux"]',
  'projects' => '[100873,1006363]',
  'startDate' => 2018-04-19,
  'endDate' => 2018-04-19,
  'granularity' => 1440
)
```

返回结果示例

查询成功

```
{
    "code": 0,
    "message": "",
    "codeDesc": "Success",
    "data": {
        "100873":[
            {
                "datetime": "2018-04-19 00:00:00",
                 "flux": 532316465152
            }
        ],
        "1006363":[
            {
                "datetime": "2018-04-19 00:00:00",
                 "flux": 602812385588
            }
        ]
    }
}
```

查询失败

```
{
    "code": 4100,
    "message": "鉴权失败，请参考文档中鉴权部分。",
    "codeDesc": "AuthFailure"
}
```

示例三 按域名维度查询

查询需求说明

查询域名 `dsa.test.qcloud.com` 在2018-04-19的流量监控数据，查询时间粒度5分钟。

GET 请求

GET 请求需要将所有参数都加在 URL 后：

```
https://dsa.api.qcloud.com/v2/index.php?
Action=GetDsaStatistics
&SecretId=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
&Timestamp=1524279600
&Nonce=123456789
&Signature=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
&metrics=["flux"]
&hosts=["dsa.test.qcloud.com"]
&startDate=2018-04-19
&endDate=2018-04-19
&granularity=5
```

POST 请求示例

POST 请求时，参数填充在 HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array (
  'Action' => 'GetDsaHostLogs',
  'SecretId' => 'SecretId',
  'Timestamp' => 1524279600,
  'Nonce' => 123456789,
  'Signature' => 'Signature',
  'metrics' => '["flux"]',
  'hosts' => '["dsa.test.qcloud.com"]',
  'startDate' => 2018-04-19,
  'endDate' => 2018-04-19,
  'granularity' => 5
)
```

返回结果示例

查询成功

```
{
    "code": 0,
    "message": "",
    "codeDesc": "Success",
    "data": {
        "dsa.test.qcloud.com":[
            {
                "datetime": "2018-04-19 00:00:00",
                "flux": 74634551
            },
            {
                "datetime": "2018-04-19 00:05:00",
                "flux": 69727465
            },
            {
                "datetime": "2018-04-19 00:10:00",
                "flux": 67541896
            },
             ......
            {
                "datetime": "2018-04-19 23:55:00",
                "flux": 80589746
            }
        ]
    }
}
```

查询失败

```
{
    "code": 4100,
    "message": "鉴权失败，请参考文档中鉴权部分。",
```

```
    "codeDesc": "AuthFailure"
}
```

带宽监控数据查询

最近更新时间：2019-12-31 14:53:13

接口描述

本接口示例用于指导带宽监控数据查询。
域名：`dsa.api.qcloud.com`
接口名：`GetDsaStatistics`

> **注意：**
> 1. 该接口调用频次上限为100次/分钟，请勿高频调用。
> 2. 可一次查询多个项目或多个域名的监控明细数据。
> 3. 支持最长查询时间跨度90天。

请求参数

以下请求参数列表仅列出了接口请求参数，正式调用时需要加上公共请求参数，详情请参见 公共请求参数 页面，此接口的 Action 字段为 `GetDsaStatistics` 。

参数	必选	类型	示例	描述
metrics	是	String	["bandwidth"]	统计指标，按JSON 格式提交 例如：["bandwidth"]
projects	否	Unsigned	[1001853]	项目 ID 列表，查看项目 ID 按 JSON 格式提交
hosts	否	String	["test.qcloud.com"]	域名列表 按 JSON 格式提交
startDate	否	String	2018-04-19	开始时间，默认为查询当日日期 格式：YY-MM-DD
endDate	否	String	2018-04-20	结束时间，默认为查询当日日期 格式：YY-MM-DD
granularity	否	Unsigned	15	查询时间粒度（分钟）

> **注意：**
> 1. 未指定查询目标时，默认按账号维度统计带宽数据。
> 2. 当指定查询项目列表时，按项目维度统计带宽数据。
> 3. 当指定查询域名列表时，按域名维度统计带宽数据。
> 4. 域名列表优先级高于项目列表，当同时指定查询项目列表和域名列表时，按指定的域名列表响应。
> 5. 当查询多个项目或多个域名时，为了防止查询参数过长导致 URL 越界，建议采用 POST 方式提交请求。

查询时间粒度规则

起止时间跨度	默认粒度	支持查询时间粒度
1 天	1	1，5，15，30，60，120，240，1440
2 ~ 3 天	15	15，30，60，120，240，1440
4 ~ 7 天	30	30，60，120，240，1440

8～90 天	60	60，120，240，1440

说明：

1. 带宽监控数据每1分钟为一个基本数据采样点。
2. 5分钟粒度的访问带宽，以1分钟粒度带宽为样本点，取5分钟内的平均带宽值。
3. 大于5分钟粒度的访问带宽，以5分钟粒度带宽为样本点，取最大带宽值。

响应参数

参数	类型	描述
code	Int	公共错误码 0：表示成功 其他值：表示失败 详情请参见 错误码 文档中的公共错误码说明
message	String	模块错误信息描述，与接口相关
codeDesc	String	英文错误信息，或业务侧错误码
data	Array或Object	详细查询数据信息，详细请参见 data 字段说明

data 字段说明

1、按账号维度查询时，data内容是一个数组，每个数组元素包含一个监控点的带宽明细数据，数组字段如下：

参数	类型	描述
datetime	String	数据时间刻度 例如：2018-04-19 01:00:00
bandwidth	Unsigned	带宽明细值，单位：bps

2、按项目维度查询时，data内容是一个对象，每个对象元素是一个项目的带宽明细数组，具体信息如下：

参数	类型	描述
项目ID	Array	项目详细带宽监控数据数组，数组字段如下： “datetime”：数据时间刻度 “bandwidth”：带宽明细值，单位：bps

3、按域名维度查询时，data内容是一个对象，每个对象元素是一个域名的带宽明细数组，具体信息如下：

参数	类型	描述
域名	Array	域名详细带宽监控数据数组，数组字段如下： “datetime”：数据时间刻度 “bandwidth”：带宽明细值，单位：bps

示例一 按账号维度查询

查询需求说明

查询账号下全部域名2018-04-19的带宽监控数据，查询时间粒度60分钟。

GET 请求

GET 请求需要将所有参数都加在 URL 后：

```
https://dsa.api.qcloud.com/v2/index.php?
Action=GetDsaStatistics
&SecretId=XXXXXXXXXXXXXXXXXXXXXXXXXXXX
&Timestamp=1524279600
&Nonce=123456789
&Signature=XXXXXXXXXXXXXXXXXXXXXXXXXXXX
&metrics=["bandwidth"]
&startDate=2018-04-19
&endDate=2018-04-19
&granularity=60
```

POST 请求示例

POST 请求时，参数填充在 HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array (
  'Action' => 'GetDsaHostLogs',
  'SecretId' => 'SecretId',
  'Timestamp' => 1524279600,
  'Nonce' => 123456789,
  'Signature' => 'Signature',
  'metrics' => '["bandwidth"]',
  'startDate' => 2018-04-19,
  'endDate' => 2018-04-19,
  'granularity' => 60
)
```

返回结果示例

查询成功

```
{
    "code": 0,
    "message": "",
    "codeDesc": "Success",
    "data": [
        {
            "datetime": "2018-04-19 00:00:00",
            "bandwidth": 589746515
        },
        {
            "datetime": "2018-04-19 01:00:00",
            "bandwidth": 489746515
        },
        {
            "datetime": "2018-04-19 02:00:00",
            "bandwidth": 375489625
        },
         ......
```

```
        {
            "datetime": "2018-04-19 23:00:00",
            "bandwidth": 589746515
        }
    ]
}
```

查询失败

```
{
    "code": 4100,
    "message": "鉴权失败，请参考文档中鉴权部分。",
    "codeDesc": "AuthFailure"
}
```

示例二 按项目维度查询

查询需求说明

同时查询两个项目在2018-04-19的峰值带宽，项目 ID 分别为100873和1006363。

GET 请求

GET 请求需要将所有参数都加在 URL 后：

```
https://dsa.api.qcloud.com/v2/index.php?
Action=GetDsaStatistics
&SecretId=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
&Timestamp=1524279600
&Nonce=123456789
&Signature=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
&metrics=["bandwidth"]
&projects=[100873,1006363]
&startDate=2018-04-19
&endDate=2018-04-19
&granularity=1440
```

POST 请求示例

POST 请求时，参数填充在 HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array (
  'Action' => 'GetDsaHostLogs',
  'SecretId' => 'SecretId',
  'Timestamp' => 1524279600,
  'Nonce' => 123456789,
  'Signature' => 'Signature',
  'metrics' => '["bandwidth"]',
  'projects' => '[100873,1006363]',
  'startDate' => 2018-04-19,
  'endDate' => 2018-04-19,
  'granularity' => 1440
```

```
)
```

返回结果示例

查询成功

```
{
    "code": 0,
    "message": "",
    "codeDesc": "Success",
    "data": {
        "100873":[
            {
                "datetime": "2018-04-19 00:00:00",
                 "bandwidth": 589746515
            }
        ],
        "1006363":[
            {
                "datetime": "2018-04-19 00:00:00",
                 "bandwidth": 264875853
            }
        ]
    }
}
```

查询失败

```
{
    "code": 4100,
    "message": "鉴权失败，请参考文档中鉴权部分。",
    "codeDesc": "AuthFailure"
}
```

示例三 按域名维度查询

查询需求说明

查询域名 dsa.test.qcloud.com 2018-04-19的带宽监控数据，查询时间粒度5分钟。

GET 请求

GET 请求需要将所有参数都加在 URL 后：

```
https://dsa.api.qcloud.com/v2/index.php?
Action=GetDsaStatistics
&SecretId=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
&Timestamp=1524279600
&Nonce=123456789
&Signature=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
&metrics=["bandwidth"]
&hosts=["dsa.test.qcloud.com"]
&startDate=2018-04-19
&endDate=2018-04-19
&granularity=5
```

POST 请求示例

POST 请求时，参数填充在 HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array (
  'Action' => 'GetDsaHostLogs',
  'SecretId' => 'SecretId',
  'Timestamp' => 1524279600,
  'Nonce' => 123456789,
  'Signature' => 'Signature',
  'metrics' => '["bandwidth"]',
  'hosts' => '["dsa.test.qcloud.com"]',
  'startDate' => 2018-04-19,
  'endDate' => 2018-04-19,
  'granularity' => 5
)
```

返回结果示例

查询成功

```
{
    "code": 0,
    "message": "",
    "codeDesc": "Success",
    "data": {
        "dsa.test.qcloud.com":[
            {
                "datetime": "2018-04-19 00:00:00",
                "bandwidth": 7634651
            },
            {
                "datetime": "2018-04-19 00:05:00",
                "bandwidth": 6727465
            },
            {
                "datetime": "2018-04-19 00:10:00",
                "bandwidth": 6754896
            },
             ......
            {
                "datetime": "2018-04-19 23:55:00",
                "bandwidth": 8589746
            }
        ]
    }
}
```

查询失败

```
{
```

```
    "code": 4100,
    "message": "鉴权失败，请参考文档中鉴权部分。",
    "codeDesc": "AuthFailure"
}
```

统计指标查询

统计指标查询

最近更新时间：2019-12-31 14:51:31

接口描述

本接口用于查询指定时间段内的域名访问统计指标，您可以利用此接口查询域名的访问情况。
域名：`dsa.api.qcloud.com`
接口名：`GetDsaHostStatistics`

> **注意：**
> 1. 该接口调用频次上限为100次/分钟，请勿高频调用。
> 2. 可一次查询多个项目或多个域名，分域名展示汇总用量数据。

统计指标说明

统计指标	指标名称	指标说明
总流量	flux	按域名维度统计查询时间段内的流量总和，单位：Byte
峰值带宽	bandwidth	按域名维度统计查询时间段内的峰值带宽，单位：bps
总请求次数	request	按域名维度统计查询时间段内的请求次数总和，单位：次数
平均访问延时	delay	按域名维度统计查询时间段内的平均访问延时，单位：ms

请求参数

以下请求参数列表仅列出了接口请求参数，正式调用时需要加上公共请求参数，详情请参见 公共请求参数 页面，此接口的 Action 字段为 `GetDsaHostStatistics` 。

参数	必选	类型	示例	描述
metrics	是	String	flux	统计指标名称，详情请参见 统计指标说明
projects	否	Unsigned	[1001853]	项目ID列表，查看项目ID 按 JSON 格式提交
hosts	否	String	test.qcloud.com	域名列表，当根据域名查询数据时使用 按 JSON 格式提交域名名称
startDate	否	String	2018-04-19	开始时间 格式：YY-MM-DD
endDate	否	String	2018-04-20	结束时间 格式：YY-MM-DD
offset	否	Unsigned	15	查询偏移个数
length	否	Unsigned	10	本次查询长度

参数说明：

- 开始时间或结束时间为空时，默认返回查询当日统计数据。
- 未指定查询目标时，默认按照账号维度查询全部域名的合并统计数据。

- 当查询多个项目或多个域名时，为了防止查询参数过长导致 URL 越界，建议采用 POST 方式提交请求。
- 当账号下域名较多时，您可以通过 offset 和 length 参数设置分段查询，查询域名按照 metrics 指定的第一个指标倒序排列。

响应参数

参数	类型	描述
code	Int	公共错误码 0：表示成功 其他值：表示失败 详情请参见 错误码说明 页面中的公共错误码说明
message	String	模块错误信息描述，与接口相关
codeDesc	String	英文错误信息，或业务侧错误码
data	Array	详细查询数据信息，详细请参见 data 字段说明

data 字段说明

参数名称	类型	描述
host	String	域名名称 例如：dsatest.qcloud.com
flux	Unsigned	总流量值
bandwidth	Unsigned	峰值带宽值
request	Unsigned	总请求次数
delay	Unsigned	平均访问时延

代码示例

配置需求示例

查询账号下全部域名2018-04-17的流量、带宽和请求次数统计数据。

GET 请求

GET 请求需要将所有参数都加在 URL 后：

```
https://dsa.api.qcloud.com/v2/index.php?
Action=GetDsaHostStatistics
&SecretId=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
&Timestamp=1524279600
&Nonce=123456789
&Signature=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
&metrics=[bandwidth]
&startDate=2018-04-17
&endDate=2018-04-17
```

> **注意：**
> 为了防止查询参数过长，导致 URL 越界，该接口默认采用 POST 方式提交查询请求。

POST 请求

POST 请求时，参数填充在 HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array (
  'Action' => 'GetDsaStatistics',
  'SecretId' => 'SecretId',
  'Timestamp' => 1524279600,
  'Nonce' => 123456789,
  'Signature' => 'Signature',
  'metrics' => '["flux","bandwidth","request"]',
  'startDate' => "2018-04-17"
  'endData' => "2018-04-17"
)
```

返回结果示例

查询成功

```
{
    "code": 0,
    "message": "",
    "codeDesc": "Success",
    "data": [{
            "host": "a.dsa.qcloud.com",
            "flux": 265412354,
            "bandwidth": 589746515,
            "request": 123456548
        },
        ......
        {
            "host": "b.dsa.qcloud.com",
            "flux": 362124545,
            "bandwidth": 749516585,
            "request": 156534248
        }
    ]
}
```

查询失败

```
{
    "code": 4100,
    "message": "鉴权失败，请参考文档中鉴权部分。",
    "codeDesc": "AuthFailure"
}
```

日志相关接口
访问日志下载

最近更新时间：2021-01-06 16:09:08

接口描述

本接口用于查询域名日志下载链接。域名：`dsa.api.qcloud.com`
接口名：`GetDsaHostLogs`

Note:

- 该接口调用频次上限为 100 次/分钟，请勿高频调用。
- 该接口最多支持同时查询 10 个域名的日志下载链接。
- 默认情况下日志按照小时粒度打包并生成下载链接。
- 日志下载链接有效期为 24 小时，过期后需重新查询下载链接。
- 若查询时间段内，某一个小时未产生访问日志，则不会生成下载链接。
- 默认按天粒度查询下载日志，最多可获取最近 40 天的访问日志。
- 日志采集需要一定时间，无法实时采集全平台访问日志，稳定时间约 2 小时左右。

请求参数

以下请求参数列表仅列出了接口请求参数，正式调用时需要加上公共请求参数，详情请参见 公共请求参数 页面，此接口的 Action 字段为 `GetDsaHostLogs` 。

参数	必选	类型	取值示例	描述
hosts	是	String	["dsatest.qcloud.com"]	域名列表，按 JSON 格式提交</br>详情请参见 代码示例
startDate	否	String	2018-04-19	开始时间</br>格式：YY-MM-DD
endDate	否	String	2018-04-19	结束时间</br>格式：YY-MM-DD

Note:

- 开始时间或结束时间为空时，默认返回当天日志下载链接。
- 当同时查询多个域名时，为了防止出现 URL 长度越界问题，建议尽量采用 POST 方式提交请求。

响应参数

参数	类型	描述
code	Int	公共错误码</br>0：表示成功</br>其他值：表示失败</br>详见请参见 错误码说明 页面的公共错误码
message	String	模块错误信息描述，与接口相关
codeDesc	String	英文错误信息，或业务侧错误码
data	Object	日志下载链接数据，分域名展示，详细请参见 data 字段说明

data 字段说明

参数	类型	描述

查询域名名称	Array	域名详细日志下载链接，详细请参见 域名详细日志下载链接字段说明

Note:
当查询多个域名时，每个域名的日志下载链接单独一个数组展示，数组名称为域名名称。
如同时查询 a.dsa.qcloud.com 和 b.dsa.qcloud.com 这两个域名的日志下载链接，则返回的 data 中，分别有数组名称为 "a.dsa.qcloud.com" 和 "b.dsa.qcloud.com" 的日志链接数组。

域名详细日志下载链接字段说明

参数	类型	描述
datetime	String	日志时间段</br>例如：2018-04-19 12:00:00
url	String	对应时间段的日志下载 URL

代码示例

配置需求示例

查询 a.dsa.qcloud.com 和 b.dsa.qcloud.com 这两个域名 2018-04-18 的日志下载链接。

GET 请求

为了防止查询参数过长，导致 URL 越界，该接口默认采用 POST 方式提交查询请求。

POST 请求

POST 请求时，参数填充在 HTTP Request-body 中，请求地址：

```
https://dsa.api.qcloud.com/v2/index.php
```

参数支持 form-data、x-www-form-urlencoded 等格式，参数数组如下：

```
array (
  'Action' => 'GetDsaHostLogs',
  'SecretId' => 'SecretId',
  'Timestamp' => 1507805426982,
  'Nonce' => 123456789,
  'Signature' => 'Signature',
  'hosts' => '["a.dsa.qcloud.com","b.dsa.qcloud.com"]',
  'startDate' => '2018-04-18',
  'endDate' => '2018-04-18'
)
```

返回结果示例

查询成功

```
{
    "code": 0,
    "message": "",
    "codeDesc": "Success",
    "data": {
        "a.dsa.qcloud.com": [
            {
                "datetime": "2018-04-18 00:00:00",
```

```
                "url": "http://dsa-log-download.cdn.qcloud.com/20180418/00/20180418-
a.dsa.qcloud.com?st=XBkcH5nGvXSCKqZcdNLORg&e=1508669326"
            },
            ......
            {
                "datetime": "2018-04-18 23:00:00",
                "url": "http://dsa-log-download.cdn.qcloud.com/20180418/23/20180418-
a.dsa.qcloud.com?st=CtkcH5VGvXDCKqZcdgLORE&e=1108623346"
            }
        ],
        "b.dsa.qcloud.com":[
            {
                "datetime": "2018-04-18 00:00:00",
                "url": "http://dsa-log-download.cdn.qcloud.com/20180418/00/20180418-
b.dsa.qcloud.com?st=XBkcH5nGvXSCKqZcdNLORg&e=1508069326"
            },
            ......
            {
                "datetime": "2018-04-18 23:00:00",
                "url": "http://dsa-log-download.cdn.qcloud.com/20180418/23/20180418-
b.dsa.qcloud.com?st=CtkcH5VGvXDCKqZcdgLORE&e=3108623346"
            }
        ]
    }
}
```

查询失败

```
{
    "code": 4100,
    "message": "鉴权失败，请参考文档中鉴权部分。",
    "codeDesc": "AuthFailure"
}
```

错误码

最近更新时间：2020-03-13 18:00:16

欢迎使用腾讯云动态加速网络（Dynamic Site Accelerator）服务！
动态加速网络（DSA）利用腾讯自研的最优链路算法及协议层优化，提供全球范围内稳定、安全的动态请求加速服务。
您可以使用本文档介绍的 API 对 DSA 服务进行相关操作，如 接入域名 、查询域名 等，具体支持的操作可参考 API 概览 。请您在使用这些接口前，确保已经充分了解 DSA 产品说明 和 计费说明 。

术语表

为了让您快速了解动态加速网络，我们对其中的一些常用术语进行了解释，如下表：

术语	中文	说明
DSA	动态加速网络	一种利用传输链路优化和传输协议优化，实现全球范围内快速、稳定、安全的动态请求加速。
CNAME	别名	域名解析中的别名记录，您可以到域名服务提供商处进行设置。
Origin	源站	用户自有的业务服务器。

快速使用入门

为了使用动态加速网络，您只需要完成以下两个步骤：

1. 添加域名您可以通过 新增加速域名 接口将域名添加入 DSA，添加成功后，DSA 会为您的域名分配对应的 CNAME ，您可以通过 查询域名列表 接口或登录 DSA 控制台查看。
   - 添加的加速域名，必须尚未添加入腾讯云 CDN 或者 DSA 服务平台；
   - 添加中国大陆区域加速的域名，必须通过工信部备案。
2. 配置 CNAME根据第一步获取的 CNAME，您需要到域名服务商处设置 CNAME ，待域名解析生效时，即开始使用动态加速网络服务。