专线接入

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 01:27:21

第 33 次发布

发布时间：2025-04-29 01:27:14

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- BgpPeer
  - **修改成员：** CloudAsn
- DirectConnect
  - 新增成员：IsThreeArch

第 32 次发布

发布时间：2025-02-24 01:26:07

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- AccessPoint
  - 新增成员：Address

第 31 次发布

发布时间：2025-01-16 01:13:53

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- CloudAttachInfo
  - 新增成员：ArRegion
- CreateCasInput
  - 新增成员：ArRegion

第 30 次发布

发布时间：2025-01-09 01:15:16

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- BgpPeer

- 新增成员：CloudAsn
- DirectConnectTunnelRoute
  - 新增成员：UpdateTime, ApplyOnTunnelEnable

第 29 次发布

发布时间：2024-12-26 01:27:35

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- DescribeAccessPoints
  - 新增入参：Filters

新增数据结构：

- PortSpecification

修改数据结构：

- AccessPoint
  - 新增成员：AvailablePortInfo

第 28 次发布

发布时间：2024-10-15 16:53:15

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- CreateCloudAttachService

新增数据结构：

- CloudAttachInfo
- CreateCasInput

第 27 次发布

发布时间：2024-09-02 01:33:14

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- CreateDirectConnect
  - 新增入参：Tags
- CreateDirectConnectTunnel
  - 新增入参：Tags

修改数据结构：

- DirectConnect
  - 新增成员：Construct, AccessPointName

第 26 次发布

发布时间：2024-06-14 01:40:38

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- DirectConnectTunnel
  - 新增成员：ShareOrNot

第 25 次发布

发布时间：2024-03-06 01:13:45

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- ModifyDirectConnectTunnelExtra
  - 新增入参：TencentIPv6Address, TencentBackupIPv6Address, CustomerIPv6Address

第 24 次发布

发布时间：2022-10-14 06:25:18

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- DirectConnectTunnelExtra
  - 新增成员：HighPrecisionBFDEnable

第 23 次发布

发布时间：2022-04-13 06:36:47

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- CreateDirectConnectTunnel
  - 新增入参：BfdEnable, NqaEnable, BfdInfo, NqaInfo

第 22 次发布

发布时间：2021-08-17 08:05:38

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- AccessPoint
  - 新增成员：AccessPointType

第 21 次发布

发布时间：2021-05-24 08:04:38

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- DirectConnectTunnelExtra
  - 新增成员：JumboEnable

第 20 次发布

发布时间：2021-04-21 08:04:26

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- ModifyDirectConnectTunnelExtra
  - 新增入参：JumboEnable

第 19 次发布

发布时间：2021-04-20 08:04:47

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- DirectConnect
  - 新增成员：LocalZone, VlanZeroDirectConnectTunnelCount, OtherVlanDirectConnectTunnelCount, MinBandwidth

第 18 次发布

发布时间：2021-04-07 08:04:13

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- ModifyDirectConnectAttribute
  - 新增入参：Bandwidth

第 17 次发布

发布时间：2021-02-23 08:04:16

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- AccessPoint
  - 新增成员：Area

第 16 次发布

发布时间：2021-01-28 08:04:02

本次发布包含了以下内容：

改善已有的文档。

新增数据结构：

- Coordinate

修改数据结构：

- AccessPoint
  - 新增成员：Coordinate, City

第 15 次发布

发布时间：2021-01-19 08:04:38

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- ModifyDirectConnectTunnelExtra
  - 新增入参：CustomerIDCRoutes

第 14 次发布

发布时间：2020-12-25 08:03:57

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- ApplyInternetAddress
- DescribeInternetAddress
- DescribeInternetAddressQuota
- DescribeInternetAddressStatistics
- DisableInternetAddress
- EnableInternetAddress
- ReleaseInternetAddress

修改接口：

- ModifyDirectConnectTunnelExtra
  - 新增入参：IPv6Enable

新增数据结构：

- InternetAddressDetail
- InternetAddressStatistics

修改数据结构：

- DirectConnectTunnelExtra
  - 新增成员：IPv6Enable, TencentIPv6Address, TencentBackupIPv6Address, BgpIPv6Status, CustomerIPv6Address

第 13 次发布

发布时间：2020-12-18 08:04:11

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- CreateDirectConnectTunnel
  - 新增入参：CloudAttachId

修改数据结构：

- DirectConnectTunnel
  - 新增成员：CloudAttachId

第 12 次发布

发布时间：2020-10-14 08:03:21

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DescribeDirectConnectTunnelExtra
- DescribePublicDirectConnectTunnelRoutes
- ModifyDirectConnectTunnelExtra

新增数据结构：

- BFDInfo
- BGPStatus
- DirectConnectTunnelExtra
- DirectConnectTunnelRoute
- NQAInfo

第 11 次发布

发布时间：2020-08-04 08:10:19

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- CreateDirectConnect
  - 新增入参：SignLaw
  - **修改入参：** Location
- DescribeDirectConnects
  - 新增出参：AllSignLaw
- ModifyDirectConnectAttribute
  - 新增入参：SignLaw

第 10 次发布

发布时间：2020-08-03 08:10:29

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- AccessPoint
  - 新增成员：AvailablePortType
- DirectConnect
  - 新增成员：SignLaw
- DirectConnectTunnel
  - 新增成员：SignLaw

第 9 次发布

发布时间：2020-06-10 08:08:58

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- DirectConnect
  - 新增成员：StartTime

第 8 次发布

发布时间：2020-04-21 08:08:32

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- ModifyDirectConnectTunnelAttribute
  - 新增入参：TencentBackupAddress

第 7 次发布

发布时间：2020-04-17 08:07:36

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- CreateDirectConnectTunnel
  - 新增入参：TencentBackupAddress

第 6 次发布

发布时间：2020-04-15 08:08:41

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- DirectConnectTunnel
  - 新增成员：TencentBackupAddress

第 5 次发布

发布时间：2020-03-20 08:06:49

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- DirectConnect
  - 新增成员：IdcCity, ChargeState
- DirectConnectTunnel
  - 新增成员：DirectConnectGatewayName, VpcName

第 4 次发布

发布时间：2020-03-16 08:08:04

本次发布包含了以下内容：

改善已有的文档。

新增数据结构：

- Tag

修改数据结构：

- DirectConnect
  - 新增成员：TagSet, AccessPointType
- DirectConnectTunnel
  - 新增成员：TagSet, NetDetectId, EnableBGPCommunity, NatType, VpcRegion, BfdEnable, AccessPointType

第 3 次发布

发布时间：2019-05-09 22:03:57

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- CreateDirectConnect

- DeleteDirectConnect
- DescribeAccessPoints
- DescribeDirectConnects
- ModifyDirectConnectAttribute

新增数据结构：

- AccessPoint
- DirectConnect

第 2 次发布

发布时间：2018-08-17 13:00:45

本次发布包含了以下内容：

改善已有的文档。

第 1 次发布

发布时间：2018-07-19 16:17:35

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- AcceptDirectConnectTunnel
- CreateDirectConnectTunnel
- DeleteDirectConnectTunnel
- DescribeDirectConnectTunnels
- ModifyDirectConnectTunnelAttribute
- RejectDirectConnectTunnel

新增数据结构：

- BgpPeer
- DirectConnectTunnel
- Filter
- RouteFilterPrefix

简介

最近更新时间：2025-04-25 01:25:37

欢迎使用腾讯云专线接入（ DC ）。专线接入提供了一种快速安全连接腾讯云与本地数据中心的方法，用户可以通过一条物理专线一次性打通位于多地域的腾讯云计算资源，实现灵活可靠的混合云部署。本文档提供的 API 供您使用请求调用的方式来操作专线接入。请确保在使用这些接口前，已充分了解 DC 产品、及其使用和收费方式。

在本文档的接口说明部分，凡出现任何参数可选范围等方面与腾讯云官网上给出的数值发生矛盾时，均以官网上给出的值为准。

注意：

- 本章节专线接入 API 接口均为最新 API 3.0 接口，后续 DC 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 API 3.0 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：专线接入 API（旧版）简介。

API 概览

最近更新时间：2024-10-15 16:53:21

专线接入相关接口

接口名称	接口功能	频率限制（次/秒）
AcceptDirectConnectTunnel	接受专用通道申请	20
ApplyInternetAddress	申请互联网地址	20
CreateCloudAttachService	创建敏捷上云服务	20
CreateDirectConnect	申请物理专线	20
CreateDirectConnectTunnel	创建专用通道	20
DeleteDirectConnect	删除物理专线	20
DeleteDirectConnectTunnel	删除专用通道	20
DescribeAccessPoints	查询物理专线接入点	20
DescribeDirectConnectTunnelExtra	查询专用通道扩展信息	20
DescribeDirectConnectTunnels	查询专用通道列表	20
DescribeDirectConnects	查询物理专线列表	20
DescribeInternetAddress	获取用户互联网公网地址信息	20
DescribeInternetAddressQuota	获取互联网公网地址配额	20
DescribeInternetAddressStatistics	获取用户互联网公网地址统计信息	20
DescribePublicDirectConnectTunnelRoutes	查询互联网通道路由列表	20
DisableInternetAddress	停用公网互联网地址	20
EnableInternetAddress	启用互联网公网地址	20
ModifyDirectConnectAttribute	修改物理专线属性	20
ModifyDirectConnectTunnelAttribute	修改专用通道属性	20
ModifyDirectConnectTunnelExtra	修改专用通道扩展信息	20
RejectDirectConnectTunnel	拒绝专用通道申请	20
ReleaseInternetAddress	释放互联网地址	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 01:16:01

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 dc.tencentcloudapi.com ，也支持指定地域域名访问，例如广州地域的域名为 dc.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 dc.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）	dc.tencentcloudapi.com
华南地区（广州）	dc.ap-guangzhou.tencentcloudapi.com
华东地区（上海）	dc.ap-shanghai.tencentcloudapi.com
华东地区（南京）	dc.ap-nanjing.tencentcloudapi.com
华北地区（北京）	dc.ap-beijing.tencentcloudapi.com
西南地区（成都）	dc.ap-chengdu.tencentcloudapi.com
西南地区（重庆）	dc.ap-chongqing.tencentcloudapi.com
港澳台地区（中国香港）	dc.ap-hongkong.tencentcloudapi.com
亚太东南（新加坡）	dc.ap-singapore.tencentcloudapi.com
亚太东南（雅加达）	dc.ap-jakarta.tencentcloudapi.com
亚太东南（曼谷）	dc.ap-bangkok.tencentcloudapi.com
亚太东北（首尔）	dc.ap-seoul.tencentcloudapi.com
亚太东北（东京）	dc.ap-tokyo.tencentcloudapi.com
美国东部（弗吉尼亚）	dc.na-ashburn.tencentcloudapi.com
美国西部（硅谷）	dc.na-siliconvalley.tencentcloudapi.com
南美地区（圣保罗）	dc.sa-saopaulo.tencentcloudapi.com
欧洲地区（法兰克福）	dc.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）	dc.ap-shanghai-fsi.tencentcloudapi.com
华南地区（深圳金融）	dc.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 01:29:05

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 dc；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********************************
```

地域列表

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

地域	取值
华南地区（广州）	ap-guangzhou

签名方法 v3

最近更新时间：2024-12-25 01:27:48

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；

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

字段名称	解释
	UTC 标准时间日期一致；service 为产品名，必须与调用的产品域名一致。此示例计算结果是 2019-02-25/cvm/tc3_request。
HashedCanonicalRequest	前述步骤拼接所得规范请求串的哈希值，计算伪代码为 Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))。此示例计算结果是 7019a55be8395899b900fb5564e4200d984910f34794a27cb3fb7d10ff6a1e84。

注意：

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

根据以上规则，示例中得到的待签名字符串如下：

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

3. 计算签名

1）计算派生签名密钥，伪代码如下：

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

派生出的密钥 SecretDate、SecretService 和 SecretSigning 是二进制的数据，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
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID*******************************
$secretId = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
$secretKey = getenv("TENCENTCLOUD_SECRET_KEY");
$host = "cvm.tencentcloudapi.com";
$service = "cvm";
$version = "2017-03-12";
$action = "DescribeInstances";
$region = "ap-guangzhou";
// $timestamp = time();
$timestamp = 1551113065;
$algorithm = "TC3-HMAC-SHA256";

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

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

```
.$timestamp."\n"
.$credentialScope."\n"
.$hashedCanonicalRequest;
echo $stringToSign.PHP_EOL;

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

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

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

Ruby

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

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

service = 'cvm'
host = 'cvm.tencentcloudapi.com'
endpoint = 'https://' + host
region = 'ap-guangzhou'
action = 'DescribeInstances'
```

```
version = '2017-03-12'
algorithm = 'TC3-HMAC-SHA256'
# timestamp = Time.now.to_i
timestamp = 1551113065
date = Time.at(timestamp).utc.strftime('%Y-%m-%d')

# ************* 步骤 1：拼接规范请求串 *************
http_request_method = 'POST'
canonical_uri = '/'
canonical_querystring = ''
canonical_headers = "content-type:application/json; charset=utf-8\nhost:#{host}\nx-tc-action:#{action.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。
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 01:27:48

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 01:25:01

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 06:27:43

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。

专线接入相关接口
接受专用通道申请

最近更新时间：2025-04-29 01:27:20

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

接受专用通道申请。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：AcceptDirectConnectTunnel。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。
DirectConnectTunnelId	是	String	专用通道ID。可以通过DescribeDirectConnectTunnels接口获取。 示例值：dcx-xxxxxxxx

3. 输出参数

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

4. 示例

示例1 接受专用通道申请

接受专用通道申请

输入示例

```
https://dc.tencentcloudapi.com/?Action=AcceptDirectConnectTunnel
&DirectConnectTunnelId=dcx-abcdefgh
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "3c140219-cfe9-470e-b241-907877d6fb03"
}
}
```

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	内部错误。
InvalidParameter	参数错误。
MissingParameter	缺少参数错误。
ResourceNotFound	资源不存在。
ResourceNotFound.DirectConnectTunnelIdIsNotExist	专用通道不存在。
UnsupportedOperation.StateConfLict	状态冲突。

申请互联网地址

最近更新时间：2025-04-25 01:25:37

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

申请互联网CIDR地址

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：ApplyInternetAddress。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。
MaskLen	是	Integer	CIDR地址掩码长度 示例值：30
AddrType	是	Integer	0:BGP类型地址 1：中国电信 2：中国移动 3：中国联通 示例值：0
AddrProto	是	Integer	0：IPv4 1:IPv6 示例值：0

3. 输出参数

参数名称	类型	描述
InstanceId	String	互联网公网地址ID 示例值：ipv4-h2k2jna1
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 申请互联网BGP IPv4地址

输入示例

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

{
"AddrType": "0",
"MaskLen": "30",
"AddrProto": "0"
}
```

输出示例

```
{
"Response": {
"InstanceId": "ipv4-h2k2jna1",
"RequestId": "24a0d7e5-4c13-49be-aa16-94f698fbef3e"
}
}
```

示例2 远程出口公网IP申请示例

输入示例

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

{
"AddrType": "1",
"MaskLen": "30",
"AddrProto": "0"
}
```

输出示例

```
{
"Response": {
"InstanceId": "ipv4-ljm17pbl",
"RequestId": "4a871804-9877-43f3-a2ba-0c213ab72af9"
}
}
```

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
LimitExceeded	超过配额限制。

申请物理专线

最近更新时间：2025-04-29 01:27:20

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

申请物理专线接入。
调用该接口时，请注意：
账号要进行实名认证，否则不允许申请物理专线；
若账户下存在欠费状态的物理专线，则不能申请更多的物理专线。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：CreateDirectConnect。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。
DirectConnectName	是	String	物理专线的名称。 示例值：北京航信物理专线1
AccessPointId	是	String	物理专线所在的接入点。 您可以通过调用DescribeAccessPoints接口获取接入点ID。 示例值：ap-cn-beijing-hx
LineOperator	是	String	提供接入物理专线的运营商。 ChinaTelecom：中国电信； ChinaMobile：中国移动； ChinaUnicom：中国联通； In-houseWiring：楼内线； ChinaOther：中国其他； InternationalOperator：境外其他。 示例值：ChinaMobile
PortType	是	String	物理专线接入端口类型，取值： 100Base-T：百兆电口； 1000Base-T（默认值）：千兆电口； 1000Base-LX：千兆单模光口（10千米）； 10GBase-T：万兆电口； 10GBase-LR（默认值）：万兆单模光口（10千米）。 示例值：1000Base-LX
CircuitCode	否	String	运营商或者服务商为物理专线提供的电路编码。 示例值：北京航信ANE0348NP

参数名称	必选	类型	描述
Location	否	String	本地数据中心的地理位置。 示例值：北京市海淀区西格玛A大厦14楼
Bandwidth	否	Integer	物理专线接入接口带宽，单位为Mbps，默认值为1000，取值范围为 [2, 10240]。 示例值：1000
RedundantDirectConnectId	否	String	冗余物理专线的ID。 示例值：dc-12345678
Vlan	否	Integer	物理专线调试VLAN。默认开启VLAN，自动分配VLAN。 示例值：100
TencentAddress	否	String	物理专线调试腾讯侧互联 IP。默认自动分配。 示例值：IP地址
CustomerAddress	否	String	物理专线调试用户侧互联 IP。默认自动分配。 示例值：客户端IP地址
CustomerName	否	String	物理专线申请者姓名。默认从账户体系获取。 示例值：张三
CustomerContactMail	否	String	物理专线申请者联系邮箱。默认从账户体系获取。 示例值：12345@qq.com
CustomerContactNumber	否	String	物理专线申请者联系号码。默认从账户体系获取。 示例值：180XXXXXXXX
FaultReportContactPerson	否	String	报障联系人。 示例值：李四
FaultReportContactNumber	否	String	报障联系电话。 示例值：181XXXXXXXX
SignLaw	否	Boolean	物理专线申请者是否签署了用户使用协议。默认已签署。 示例值：是
Tags.N	否	Array of Tag	标签键值对

3. 输出参数

参数名称	类型	描述
DirectConnectIdSet	Array of String	物理专线的ID。 示例值：["dc-abcdefgh"]
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 申请物理专线1

申请物理专线，接入点为北京航信，中国移动线路，云端端口为千兆单模光口（1000Base-LX）。

输入示例

```
https://dc.tencentcloudapi.com/?Action=CreateDirectConnect
&DirectConnectName=北京航信物理专线1
&AccessPointId=ap-cn-beijing-hx
&LineOperator=ChinaMobile
&CircuitCode=北京航信ANE0348NP
&Location=北京市海淀区西格玛A大厦14楼
&PortType=1000Base-LX
&Bandwidth=1000
&CustomerName=张三
&CustomerContactMail=12345@qq.com
&CustomerContactNumber=18812345678
&<公共请求参数>
```

输出示例

```
{
"Response": {
"DirectConnectIdSet": [
"dc-abcdefgh"
],
"RequestId": "24a0d7e5-4c13-49be-aa16-94f698fbef3e"
}
}
```

示例2 申请物理专线2

申请物理专线，接入点为南山，中国移动线路，云端端口为千兆单模光口（1000Base-LX），有冗余专线。

输入示例

```
https://dc.tencentcloudapi.com/?Action=CreateDirectConnect
&DirectConnectName=物理专线1
&AccessPointId=ap-cn-shenzhen-ns-A
&LineOperator=ChinaMobile
&CircuitCode=深圳南山ANE0348NP
&Location=广东省深圳市南山区科技园A大厦14楼
&PortType=1000Base-LX
&Bandwidth=1000
&RedundantDirectConnectId=dc-abcdedf
&Vlan=100
&TencentAddress=172.168.1.1/30
&CustomerAddress=172.168.1.2/30
&CustomerName=张三
&CustomerContactMail=12345@qq.com
&CustomerContactNumber=18812345678
&<公共请求参数>
```

输出示例

```
{
"Response": {
"DirectConnectIdSet": [
"dc-abcdefgh"
],
"RequestId": "24a0d7e5-4c13-49be-aa16-94f698fbef3e"
}
}
```

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	内部错误。
LimitExceeded	超过配额限制。
LimitExceeded.DirectConnectLimitExceeded	物理专线数已达上限。
ResourceNotFound	资源不存在。
UnauthorizedOperation	未授权操作。
UnsupportedOperation	操作不支持。

创建专用通道

最近更新时间：2025-04-25 01:25:37

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

创建专用通道。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：CreateDirectConnectTunnel。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。
DirectConnectId	是	String	物理专线ID，例如：dc-kd7d06of。 示例值：dc-abcdefgh
DirectConnectTunnelName	是	String	专用通道名称。 示例值：Test
DirectConnectOwnerAccount	否	String	物理专线owner，缺省为当前客户（物理专线 owner） 共享专线时这里需要填写共享专线的开发商账号 ID。 示例值：12325464
NetworkType	否	String	网络类型，枚举：VPC、CCN、NAT；默认为VPC。VPC：私有网络；CCN：云联网；NAT：NAT网络）。 示例值：VPC
NetworkRegion	否	String	网络地域。 示例值：ap-guangzhou
VpcId	否	String	私有网络统一ID，在NetworkType为VPC时必填，且与专线网关所属的VPCID一致；NetworkType为其它组网类型时可不填，内部会统一处理。 示例值：vpc-abcdefgh
DirectConnectGatewayId	否	String	专线网关ID，例如 dcg-d545ddf。 示例值：dcg-abcdefgh
Bandwidth	否	Integer	专线带宽，单位：Mbps；默认是物理专线带宽值。 示例值：100
RouteType	否	String	路由类型，枚举：BGP、STATIC；默认为BGP 。（BGP ： BGP 路由；STATIC：静态）。

参数名称	必选	类型	描述
			示例值：BGP
BgpPeer	否	BgpPeer	BgpPeer，用户侧bgp信息，包括Asn和AuthKey。
RouteFilterPrefixes.N	否	Array of RouteFilterPrefix	静态路由，用户IDC的网段地址。
Vlan	否	Integer	vlan，范围：0～3000。 0：不开启子接口，默认值是非0。 示例值：100
TencentAddress	否	String	TencentAddress，腾讯侧互联 IP。 示例值：192.168.1.2/30
CustomerAddress	否	String	CustomerAddress，用户侧互联 IP。 示例值：192.168.1.1/30
TencentBackupAddress	否	String	TencentBackupAddress，腾讯侧备用互联 IP。 示例值：192.168.1.1/30
CloudAttachId	否	String	高速上云服务ID。 示例值：cas-xxxxxxxx
BfdEnable	否	Integer	是否开启BFD。 示例值：0
NqaEnable	否	Integer	是否开启NQA。 示例值：0
BfdInfo	否	BFDInfo	BFD配置信息。
NqaInfo	否	NQAInfo	NQA配置信息。
Tags.N	否	Array of Tag	标签键值对

3. 输出参数

参数名称	类型	描述
DirectConnectTunnelIdSet	Array of String	专用通道ID。 示例值：["dcx-abcdefgh"]
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 创建云交换服务通道

创建云交换服务通道

输入示例

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

```
{
"DirectConnectId": "dc-juystu9z",
"DirectConnectTunnelName": "testvif",
"NetworkType": "CCN",
"NetworkRegion": "ap-guangzhou",
"DirectConnectGatewayId": "dcg-du8uzrih",
"Bandwidth": 100,
"RouteType": "BGP",
"Vlan": 1001
}
```

输出示例

```
{
"Response": {
"DirectConnectTunnelIdSet": [
"dcx-inx9ty4u"
],
"RequestId": "4539586b-d0d7-49bb-b351-2036da9ac277"
}
}
```

示例2 BGP路由模式和云联网专用通道

BGP路由模式和云联网专用通道

输入示例

```
https://dc.tencentcloudapi.com/?Action=CreateDirectConnectTunnel
&DirectConnectId=dc-abcdefgh
&DirectConnectTunnelName=Test
&NetworkType=CCN
&NetworkRegion=ap-guangzhou
&DirectConnectGatewayId=dcg-abcdefgh
&Bandwidth=100
&RouteType=BGP
&Vlan=100
&TencentAddress=192.168.1.2/30
&CustomerAddress=192.168.1.1/30
&BgpPeer.Asn=65128
&BgpPeer.AuthKey=abcdefg
&<公共请求参数>
```

输出示例

```
{
"Response": {
"DirectConnectTunnelIdSet": [
```

```
"dcx-abcdefgh"
],
"RequestId": "24a0d7e5-4c13-49be-aa16-94f698fbef3e"
}
}
```

示例3 BGP路由模式和共享专线的专用通道

BGP路由模式和共享专线的专用通道

输入示例

```
https://dc.tencentcloudapi.com/?Action=CreateDirectConnectTunnel
&DirectConnectId=dc-abcdefgh
&DirectConnectTunnelName=Test
&DirectConnectOwnerAccount=240791248
&NetworkType=VPC
&NetworkRegion=ap-guangzhou
&VpcId=vpc-abcdefgh
&DirectConnectGatewayId=dcg-abcdefgh
&Bandwidth=100
&RouteType=BGP
&Vlan=100
&TencentAddress=192.168.1.2/30
&CustomerAddress=192.168.1.1/30
&BgpPeer.Asn=65128
&BgpPeer.AuthKey=abcdefg
&<公共请求参数>
```

输出示例

```
{
"Response": {
"DirectConnectTunnelIdSet": [
"dcx-abcdefgh"
],
"RequestId": "24a0d7e5-4c13-49be-aa16-94f698fbef3e"
}
}
```

示例4 STATIC路由模式和黑石VPC的专用通道

STATIC路由模式和黑石VPC的专用通道

输入示例

```
https://dc.tencentcloudapi.com/?Action=CreateDirectConnectTunnel
&DirectConnectId=dc-abcdefgh
&DirectConnectTunnelName=Test
&NetworkType=BMVPC
```

```
&NetworkRegion=ap-guangzhou
&VpcId=vpc-abcdefgh
&Bandwidth=100
&RouteType=STATIC
&Vlan=100
&TencentAddress=192.168.1.2/30
&CustomerAddress=192.168.1.1/30
&RouteFilterPrefixes.0.Cidr=192.168.0.0/24
&RouteFilterPrefixes.1.Cidr=192.168.0.0/24
&RouteFilterPrefixes.2.Cidr=192.168.0.0/24
&<公共请求参数>
```

输出示例

```
{
"Response": {
"DirectConnectTunnelIdSet": [
"dcx-abcdefgh"
],
"RequestId": "24a0d7e5-4c13-49be-aa16-94f698fbef3e"
}
}
```

示例5 BGP路由模式和私有网络VPC的专用通道

BGP路由模式和私有网络VPC的专用通道

输入示例

```
https://dc.tencentcloudapi.com/?Action=CreateDirectConnectTunnel
&DirectConnectId=dc-abcdefgh
&DirectConnectTunnelName=Test
&NetworkType=VPC
&NetworkRegion=ap-guangzhou
&VpcId=vpc-abcdefgh
&DirectConnectGatewayId=dcg-abcdefgh
&Bandwidth=100
&RouteType=BGP
&Vlan=100
&TencentAddress=192.168.1.2/30
&CustomerAddress=192.168.1.1/30
&BgpPeer.Asn=65128
&BgpPeer.AuthKey=abcdefg
&<公共请求参数>
```

输出示例

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

```
"DirectConnectTunnelIdSet": [
"dcx-abcdefgh"
],
"RequestId": "24a0d7e5-4c13-49be-aa16-94f698fbef3e"
}
}
```


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	内部错误。
InvalidParameter	参数错误。
InvalidParameter.AddressError	互联IP错误。
InvalidParameter.DirectConnectIdIsNotUin	物理专线不属于该账号。
InvalidParameter.UinIsNotExist	该账号ID不存在。
InvalidParameter.VlanConflict	vlan冲突。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.VlanConfLict	vlan冲突。

错误码	描述
LimitExceeded	超过配额限制。
LimitExceeded.DirectConnectLimitExceeded	物理专线数已达上限。
LimitExceeded.DirectConnectTunnelLimitExceeded	物理专线的专用通道数已达上限。
MissingParameter	缺少参数错误。
ResourceInUse.DcVpcIsExist	物理专线的vpc已经存在。
ResourceUnavailable.InsufficientBalance	对不起您的账号已欠费，欠费状态下无法开通产品，请您先行充值。
UnsupportedOperation	操作不支持。
UnsupportedOperation.CrossBorderDirectConnectTunnel	不允许创建跨境专用通道，请您联系我们。

删除物理专线

最近更新时间：2025-04-25 01:25:37

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

删除物理专线。只能删除处于已连接状态的物理专线。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DeleteDirectConnect。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。
DirectConnectId	是	String	物理专线的ID。 示例值：dc-xxxxxxxx

3. 输出参数

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

4. 示例

示例1 删除物理专线

删除物理专线

输入示例

```
https://dc.tencentcloudapi.com/?Action=DeleteDirectConnect
&DirectConnectId=dc-abcdefgh
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "3c140219-cfe9-470e-b241-907877d6fb03"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation	操作失败。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameter.DirectConnectIdIsNotUin	物理专线不属于该账号。
InvalidParameterValue	参数取值错误。
ResourceNotFound	资源不存在。
UnauthorizedOperation	未授权操作。
UnsupportedOperation	操作不支持。
UnsupportedOperation.StateConfLict	状态冲突。

删除专用通道

最近更新时间：2025-04-25 01:25:37

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

删除专用通道。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DeleteDirectConnectTunnel。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。
DirectConnectTunnelId	是	String	专用通道ID。 示例值：dcx-xxxxxxxx

3. 输出参数

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

4. 示例

示例1 删除专用通道

删除专用通道

输入示例

```
https://dc.tencentcloudapi.com/?Action=DeleteDirectConnectTunnel
&DirectConnectTunnelId=dcx-abcdefgh
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "3c140219-cfe9-470e-b241-907877d6fb03"
```

```
}
}
```

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	内部错误。
InvalidParameter	参数错误。
ResourceNotFound	资源不存在。
ResourceNotFound.DirectConnectTunnelIdIsNotExist	专用通道不存在。
UnsupportedOperation	操作不支持。
UnsupportedOperation.StateConfLict	状态冲突。

查询物理专线接入点

最近更新时间：2025-04-29 01:27:19

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

查询物理专线接入点。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeAccessPoints。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。
RegionId	否	String	接入点所在的地域。你可以通过调用DescribeRegions接口获取地域ID。 示例值：ap-guangzhou
Offset	否	Integer	偏移量，默认为0。 示例值：0
Limit	否	Integer	返回数量，默认为20，最大值为100。 示例值：100
Filters.N	否	Array of Filter	过滤参数，支持：access-point-id、isp 示例值：["Name": "access-point-id", "Values": ["ap-beijing-a-kc"]]

3. 输出参数

参数名称	类型	描述
AccessPointSet	Array of AccessPoint	接入点信息。
TotalCount	Integer	接入点总数量。 示例值：100
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 获取全量接入点信息

获取全量接入点信息。

输入示例

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

输出示例


```
{
"Response": {
"AccessPointSet": [
{
"AccessPointName": "新加坡-C-泰戈尔",
"AccessPointId": "ap-singapore-c-tagore",
"City": "新加坡",
"Area": "其它",
"RegionId": "ap-singapore",
"Location": "新加坡Dodid泰戈尔AC",
"Address": "71 Tagore Ln,Singapore 787496",
"Coordinate": {
"Lat": 1.3885116,
"Lng": 103.8277551
},
"AccessPointType": "VXLAN",
"LineOperator": [
"InternationalOperator"
],
"AvailablePortType": [
"1000BASE-LX",
"1000BASE-ZX",
"10GBASE-LR",
"10GBASE-ZR",
"100GBASE-LR4L",
"100GBASE-LR4",
"100GBASE-40KM",
"QSFPDD-400G-FR4",
"QSFPDD-400G-LR4"
],
"AvailablePortInfo": [
{
"InternationalName": "1000BASE-LX",
"Specification": 1000,
"PortType": "X"
},
{
"InternationalName": "1000BASE-ZX",
"Specification": 1000,
"PortType": "X"
},
```

```
{
"InternationalName": "10GBASE-LR",
"Specification": 10000,
"PortType": "X"
},
{
"InternationalName": "10GBASE-ZR",
"Specification": 10000,
"PortType": "X"
},
{
"InternationalName": "100GBASE-LR4L",
"Specification": 100000,
"PortType": "X"
},
{
"InternationalName": "100GBASE-LR4",
"Specification": 100000,
"PortType": "X"
},
{
"InternationalName": "100GBASE-40KM",
"Specification": 100000,
"PortType": "X"
},
{
"InternationalName": "QSFPDD-400G-FR4",
"Specification": 400000,
"PortType": "X"
},
{
"InternationalName": "QSFPDD-400G-LR4",
"Specification": 400000,
"PortType": "X"
}
],
"State": "AVAILABLE"
}
],
"RequestId": "b6aa097b-3cd9-4c79-bf41-bb0d2427ffa2",
"TotalCount": 98
}
}
```


示例2 获取指定地域接入点信息

获取指定地域接入点信息。

输入示例

```
https://dc.tencentcloudapi.com/?Action=DescribeAccessPoints
&RegionId=ap-chongqing
&<公共请求参数>
```

输出示例


```
{
"Response": {
"AccessPointSet": [
{
"AccessPointName": "重庆-A-泰和",
"AccessPointId": "ap-chongqing-a-th",
"City": "重庆",
"Area": "西南",
"RegionId": "ap-chongqing",
"Location": "重庆腾讯泰和DC",
"Address": "重庆市北碚区水土镇高新技术产业园泰和路777号",
"Coordinate": {
"Lat": 29.790833,
"Lng": 106.523072
},
"AccessPointType": "VXLAN",
"LineOperator": [
"ChinaTelecom",
"ChinaMobile",
"ChinaUnicom",
"In-houseWiring",
"ChinaOther"
],
"AvailablePortType": [
"1000BASE-LX",
"1000BASE-T",
"1000BASE-ZX",
"10GBASE-LR",
"10GBASE-ZR",
"100GBASE-LR4L",
"100GBASE-LR4",
"100GBASE-40KM",
"QSFPDD-400G-FR4",
"QSFPDD-400G-LR4"
],
"AvailablePortInfo": [
{
"InternationalName": "1000BASE-LX",
"Specification": 1000,
"PortType": "X"
},
{
"InternationalName": "1000BASE-T",
```

```
"Specification": 1000,
"PortType": "T"
},
{
"InternationalName": "1000BASE-ZX",
"Specification": 1000,
"PortType": "X"
},
{
"InternationalName": "10GBASE-LR",
"Specification": 10000,
"PortType": "X"
},
{
"InternationalName": "10GBASE-ZR",
"Specification": 10000,
"PortType": "X"
},
{
"InternationalName": "100GBASE-LR4L",
"Specification": 100000,
"PortType": "X"
},
{
"InternationalName": "100GBASE-LR4",
"Specification": 100000,
"PortType": "X"
},
{
"InternationalName": "100GBASE-40KM",
"Specification": 100000,
"PortType": "X"
},
{
"InternationalName": "QSFPDD-400G-FR4",
"Specification": 400000,
"PortType": "X"
},
{
"InternationalName": "QSFPDD-400G-LR4",
"Specification": 400000,
"PortType": "X"
}
],
"State": "AVAILABLE"
}
],
"RequestId": "b6aa097b-3cd9-4c79-bf41-bb0d2427ffa3",
"TotalCount": 3
```

```
}
}
```

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	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
MissingParameter	缺少参数错误。
ResourceNotFound	资源不存在。
ResourceUnavailable	资源不可用。
UnauthorizedOperation	未授权操作。
UnsupportedOperation	操作不支持。

查询专用通道扩展信息

最近更新时间：2025-04-25 01:25:36

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

查询专用通道扩展信息。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeDirectConnectTunnelExtra。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。
DirectConnectTunnelId	是	String	专用通道ID。 示例值：dcx-xxxxxxxx

3. 输出参数

参数名称	类型	描述
DirectConnectTunnelExtra	DirectConnectTunnelExtra	专用通道扩展信息。 示例值：{}
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 查询专用通道扩展信息

查询专用通道扩展信息

输入示例

```
https://dc.tencentcloudapi.com/?Action=DescribeDirectConnectTunnelExtra
&DirectConnectTunnelId=dcx-047zz5e6
&<公共请求参数>
```

输出示例


```
{
"Response": {
"DirectConnectTunnelExtra": {
"OwnerAccount": "100001332514",
"DirectConnectOwnerAccount": "100001332514",
"DirectConnectId": "dc-n6c9vvv3",
"SignLaw": true,
"DirectConnectTunnelId": "dcx-047zz5e6",
"DirectConnectTunnelName": "DCXCCNVxlanBgpEcmpTestautotestdcxtwo",
"State": "AVAILABLE",
"VpcId": "",
"NetworkRegion": "ap-chongqing",
"VpcRegion": "cq",
"DirectConnectGatewayId": "dcg-meljxc9n",
"Bandwidth": 100,
"Vlan": 2432,
"TencentAddress": "192.168.0.3/29",
"CustomerAddress": "192.168.0.1/29",
"CreatedTime": "2020-09-22T00:00:00+00:00",
"NetDetectId": "",
"EnableBGPCommunity": false,
"NatType": 0,
"BfdEnable": 0,
"AccessPointType": "VXLAN",
"DirectConnectGatewayName": "",
"VpcName": "",
"NqaEnable": 0,
"BfdInfo": {
"ProbeFailedTimes": -1,
"Interval": -1
},
"NqaInfo": {
"ProbeFailedTimes": -1,
"Interval": -1,
"DestinationIp": "0.0.0.0"
},
"IPv6Enable": 0,
"PublicAddresses": [],
"JumboEnable": 0,
"HighPrecisionBFDEnable": 0,
"TencentBackupAddress": "192.168.0.2/29",
"NetworkType": "CCN",
"RouteType": "BGP",
"BgpPeer": {
"Asn": 65120,
"AuthKey": "tencent"
},
"RouteFilterPrefixes": [],
```

```
"BgpStatus": {
"TencentAddressBgpState": "Established",
"TencentBackupAddressBgpState": "Connect"
},
"TencentIPv6Address": "",
"TencentBackupIPv6Address": "",
"CustomerIPv6Address": "",
"BgpIPv6Status": {
"TencentAddressBgpState": "",
"TencentBackupAddressBgpState": ""
}
},
"RequestId": "8ae32da8-db96-400f-908e-0de2c89e96ea"
}
}
```


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	内部错误。
ResourceNotFound	资源不存在。

错误码	描述
ResourceNotFound.DirectConnectTunnelIdIsNotExist	专用通道不存在。

查询专用通道列表

最近更新时间：2025-04-25 01:25:36

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

查询专用通道列表。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeDirectConnectTunnels。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。
Filters.N	否	Array of Filter	过滤条件: 参数不支持同时指定DirectConnectTunnelIds和Filters。 direct-connect-tunnel-name, 专用通道名称。 direct-connect-tunnel-id, 专用通道实例ID，如：dcx-abcdefgh。 direct-connect-id, 物理专线实例ID，如：dc-abcdefgh。
DirectConnectTunnelIds.N	否	Array of String	专用通道ID数组。 示例值：["dcx-abab1212"]
Offset	否	Integer	偏移量，默认为0。 示例值：0
Limit	否	Integer	返回数量，默认为20，最大值为100。 示例值：20

3. 输出参数

参数名称	类型	描述
DirectConnectTunnelSet	Array of DirectConnectTunnel	专用通道列表。
TotalCount	Integer	专用通道总数量。 示例值：100
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BGP路由的专用通道

查询BGP路由的专用通道。

输入示例

```
https://dc.tencentcloudapi.com/?Action=DescribeDirectConnectTunnels
&Filters.0.Name=direct-connect-tunnel-id
&Filters.0.Values.0=dcx-047zz5e6
&Limit=20
&Offset=0
&<公共请求参数>
```

输出示例


```
{
"Response": {
"DirectConnectTunnelSet": [
{
"OwnerAccount": "100001332514",
"DirectConnectOwnerAccount": "100001332514",
"DirectConnectId": "dc-n6c9vvv3",
"SignLaw": true,
"DirectConnectTunnelId": "dcx-047zz5e6",
"DirectConnectTunnelName": "DCXCCNVxlanBgpEcmpTestautotestdcxtwo",
"State": "AVAILABLE",
"VpcId": "",
"NetworkRegion": "ap-chongqing",
"VpcRegion": "cq",
"DirectConnectGatewayId": "dcg-meljxc9n",
"Bandwidth": 100,
"Vlan": 2432,
"TencentAddress": "192.168.0.3/29",
"CustomerAddress": "192.168.0.1/29",
"CreatedTime": "2020-09-22T00:00:00+00:00",
"NetDetectId": "",
"EnableBGPCommunity": false,
"NatType": 0,
"BfdEnable": 0,
"AccessPointType": "VXLAN",
"DirectConnectGatewayName": "",
"VpcName": "",
"ShareOrNot": 0,
"TencentBackupAddress": "192.168.0.2/29",
"NetworkType": "CCN",
"CloudAttachId": "",
"RouteType": "BGP",
"BgpPeer": {
"Asn": 65120,
```

```
"AuthKey": "tencent"
},
"RouteFilterPrefixes": [],
"TagSet": []
}
],
"RequestId": "8ae32da8-db96-400f-908e-0de2c89e96ea",
"TotalCount": 1
}
}
```


示例2 查询STATIC路由的专用通道

查询STATIC路由的专用通道。

输入示例

```
https://dc.tencentcloudapi.com/?Action=DescribeDirectConnectTunnels
&Filters.0.Name=direct-connect-tunnel-id
&Filters.0.Values.0=dcx-ltcfypom
&Limit=20
&Offset=0
&<公共请求参数>
```

输出示例


```
{
"Response": {
"DirectConnectTunnelSet": [
{
"OwnerAccount": "100001332514",
"DirectConnectOwnerAccount": "100001332514",
"DirectConnectId": "dc-n6c9vvv3",
"SignLaw": true,
"DirectConnectTunnelId": "dcx-ltcfypom",
"DirectConnectTunnelName": "DCXVPCVxlanStaticEcmpTestautotestdcxone",
"State": "AVAILABLE",
"VpcId": "vpc-q1cy5hgx",
"NetworkRegion": "ap-chongqing",
"VpcRegion": "cq",
"DirectConnectGatewayId": "dcg-f5ucfdg3",
"Bandwidth": 100,
"Vlan": 2414,
"TencentAddress": "192.168.0.3/29",
"CustomerAddress": "192.168.0.1/29",
"CreatedTime": "2020-09-22T00:00:00+00:00",
"NetDetectId": "",
"EnableBGPCommunity": false,
"NatType": 0,
"BfdEnable": 0,
```

```
"AccessPointType": "VXLAN",
"DirectConnectGatewayName": "",
"VpcName": "",
"ShareOrNot": 0,
"TencentBackupAddress": "",
"NetworkType": "VPC",
"CloudAttachId": "",
"RouteType": "STATIC",
"RouteFilterPrefixes": [
{
"Cidr": "19.244.99.224/32"
}
],
"BgpPeer": {
"Asn": -1,
"AuthKey": ""
},
"TagSet": []
}
],
"RequestId": "8ae32da8-db96-400f-908e-0de2c89e96ea",
"TotalCount": 1
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
AuthFailure.UnauthorizedOperation	CAM签名/鉴权错误，未授权的操作。
InternalError	内部错误。
ResourceNotFound	资源不存在。
ResourceNotFound.DirectConnectTunnelIdIsNotExist	专用通道不存在。

查询物理专线列表

最近更新时间：2025-04-29 01:27:19

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

查询物理专线列表。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeDirectConnects。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。
Filters.N	否	Array of Filter	过滤条件。direct-connect-id：物理专线ID，states：物理专线状态（AVAILABLE-就绪，PENDING-申请中，REJECTED-申请被拒绝，PENDINGPAY-待付款，PAID-付款完成，BUILDING-建设中，STOPED-建设终止，DELETED-删除完成）。
DirectConnectIds.N	否	Array of String	物理专线 ID数组。 示例值：["dc-abab1212"]
Offset	否	Integer	偏移量，默认为0。 示例值：0
Limit	否	Integer	返回数量，默认为20，最大值为100。 示例值：100

3. 输出参数

参数名称	类型	描述
DirectConnectSet	Array of DirectConnect	物理专线列表。
TotalCount	Integer	符合物理专线列表数量。 示例值：100
AllSignLaw	Boolean	用户名下物理专线是否都签署了用户协议。 示例值：true
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 查询物理专线列表-1

使用Filter进行筛选，用direct-connect-name进行筛选。

输入示例

```
https://dc.tencentcloudapi.com/?Action=DescribeDirectConnects
&Filters.0.Name=direct-connect-name
&Filters.0.Values.0=专线接入
&<公共请求参数>
```

输出示例


```
{
"Response": {
"DirectConnectSet": [
{
"DirectConnectId": "dc-c3hbbsw9",
"DirectConnectName": "专线接入测试",
"RedundantDirectConnectId": "",
"State": "AVAILABLE",
"ChargeState": "NORMAL",
"AccessPointId": "ap-cn-guangzhou-kxc",
"AccessPointName": "广州科丰路",
"AccessPointType": "VXLAN",
"Bandwidth": 10000,
"LineOperator": "ChinaOther",
"Construct": 1,
"Location": "广东省广州市华新园0602-A01机架7号位",
"PortType": "10GBase-LR",
"CreatedTime": "2020-09-22T00:00:00+00:00",
"StartTime": "2019-09-12 17:09:07",
"CircuitCode": "唐镇0103-A09-PL01-1",
"Vlan": 100,
"CustomerName": "张三",
"CustomerContactMail": "zhangsan@tencent.com",
"CustomerContactNumber": "13888888888",
"FaultReportContactPerson": "李四",
"FaultReportContactNumber": "15999999999",
"IdcCity": "广东省广州市华新园0602-A01机架7号位",
"SignLaw": false,
"LocalZone": false,
"VlanZeroDirectConnectTunnelCount": 0,
"OtherVlanDirectConnectTunnelCount": 0,
"MinBandwidth": 0,
"ExpiredTime": "2024-09-22T00:00:00+00:00",
"EnabledTime": "2020-09-22T00:00:00+00:00",
"TencentAddress": "192.168.1.2/24",
"CustomerAddress": "192.168.1.1/24",
```

```
"ChargeType": "NON_RECURRING_CHARGE",
"IsThreeArch": false,
"TagSet": []
}
],
"RequestId": "edafc0e0-3a01-4ac3-848d-e402ff04c1f3",
"TotalCount": 1,
"AllSignLaw": false
}
}
```


示例2 查询物理专线列表-2

指定DirectConnectIds查询物理专线列表。

输入示例

```
https://dc.tencentcloudapi.com/?Action=DescribeDirectConnects
&DirectConnectIds.0=dc-c3hbbsw9
&<公共请求参数>
```

输出示例


```
{
"Response": {
"DirectConnectSet": [
{
"DirectConnectId": "dc-c3hbbsw9",
"DirectConnectName": "专线接入测试",
"RedundantDirectConnectId": "",
"State": "AVAILABLE",
"ChargeState": "NORMAL",
"AccessPointId": "ap-cn-guangzhou-kxc",
"AccessPointName": "广州科丰路",
"AccessPointType": "VXLAN",
"Bandwidth": 10000,
"LineOperator": "ChinaOther",
"Construct": 1,
"Location": "广东省广州市华新园0602-A01机架7号位",
"PortType": "10GBase-LR",
"CreatedTime": "2020-09-22T00:00:00+00:00",
"StartTime": "2019-09-12 17:09:07",
"CircuitCode": "唐镇0103-A09-PL01-1",
"Vlan": 100,
"CustomerName": "张三",
"CustomerContactMail": "zhangsan@tencent.com",
"CustomerContactNumber": "13888888888",
"FaultReportContactPerson": "李四",
"FaultReportContactNumber": "15999999999",
"IdcCity": "广东省广州市华新园0602-A01机架7号位",
```

```
"SignLaw": false,
"LocalZone": false,
"VlanZeroDirectConnectTunnelCount": 0,
"OtherVlanDirectConnectTunnelCount": 0,
"MinBandwidth": 0,
"ExpiredTime": "2024-09-22T00:00:00+00:00",
"EnabledTime": "2020-09-22T00:00:00+00:00",
"TencentAddress": "192.168.1.2/24",
"CustomerAddress": "192.168.1.1/24",
"ChargeType": "NON_RECURRING_CHARGE",
"IsThreeArch": false,
"TagSet": []
}
],
"RequestId": "edafc0e0-3a01-4ac3-848d-e402ff04c1f3",
"TotalCount": 1,
"AllSignLaw": false
}
}
```


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	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
ResourceNotFound	资源不存在。
UnauthorizedOperation	未授权操作。
UnsupportedOperation	操作不支持。

获取用户互联网公网地址信息

最近更新时间：2025-04-25 01:25:36

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

获取用户互联网公网地址信息

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeInternetAddress。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。
Offset	否	Integer	偏移量，默认为0 示例值：0
Limit	否	Integer	返回数量，默认为20，最大值100 示例值：20
Filters.N	否	Array of Filter	过滤条件： • AddrType，地址类型。0：BGP 1；1: 电信；2：移动；3：联通 • AddrProto，地址类型。0：IPv4；1:IPv6 • Status，地址状态。0：使用中；1：已停用； 2：已退还 • Subnet，互联网公网地址。数组 • InstanceIds，互联网公网地址ID。数组

3. 输出参数

参数名称	类型	描述
TotalCount	Integer	互联网公网地址数量 示例值：1
Subnets	Array of InternetAddressDetail	互联网公网地址列表
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId ）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 通过远程出口过滤用户IP段

通过远程出口过滤用户IP段

输入示例

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

{
"Filters": [
{
"Values": [
"nj-1"
],
"Name": "RemoteArea"
}
]
}
```

输出示例

```
{
"Response": {
"Subnets": [
{
"Subnet": "2402:4e00:4020::",
"Region": "nj",
"ApplyTime": "2022-08-17 10:44:02",
"StopTime": "00-00-00 00:00:00",
"ReleaseTime": "00-00-00 00:00:00",
"Status": 1,
"AddrType": 0,
"MaskLen": 64,
"ReserveTime": 8,
"InstanceId": "ipv6-0yiowoa9",
"AppId": 251009028,
"AddrProto": 1
},
{
"Subnet": "10.10.10.0",
"Region": "nj",
"ApplyTime": "2022-08-16 20:24:54",
"StopTime": "00-00-00 00:00:00",
"ReleaseTime": "00-00-00 00:00:00",
"Status": 1,
"AddrType": 1,
```

```
"MaskLen": 30,
"ReserveTime": 8,
"InstanceId": "ipv4-ljm17pbl",
"AppId": 251009028,
"AddrProto": 0
}
],
"RequestId": "eee7d064-e3ae-4020-8db9-dc0cf245ccd2",
"TotalCount": 2
}
}
```


示例2 获取用户申请的公网地址信息

获取用户申请的公网地址信息

输入示例

```
https://dc.tencentcloudapi.com/?Action=DescribeInternetAddress
&Limit=20
&Offset=1
&<公共请求参数>
```

输出示例


```
{
"Response": {
"TotalCount": 1,
"Subnets": [
{
"Status": 0,
"Subnet": "220.110.1.0",
"MaskLen": 30,
"AddrType": 0,
"AppId": 251010426,
"InstanceId": "ipv4-qmda2nqv",
"AddrProto": 0,
"StopTime": "2020-09-22 00:00:00",
"Region": "gz",
"ApplyTime": "2020-09-22 00:00:00",
"ReleaseTime": "2020-09-22 00:00:00",
"ReserveTime": 8
}
],
"RequestId": "24a0d7e5-4c13-49be-aa16-94f698fbef3e"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

获取互联网公网地址配额

最近更新时间：2025-04-25 01:25:36

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

获取用户互联网公网地址配额

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeInternetAddressQuota。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。

3. 输出参数

参数名称	类型	描述
Ipv6PrefixLen	Integer	IPv6互联网公网允许的最小前缀长度 示例值：64
Ipv4BgpQuota	Integer	BGP类型IPv4互联网地址配额 示例值：64
Ipv4OtherQuota	Integer	非BGP类型IPv4互联网地址配额 示例值：0
Ipv4BgpNum	Integer	BGP类型IPv4互联网地址已使用数量 示例值：8
Ipv4OtherNum	Integer	非BGP类型互联网地址已使用数量 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 获取用户互联网公网地址配额信息

输入示例

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

输出示例

```
{
"Response": {
"Ipv4BgpQuota": 256,
"Ipv4OtherQuota": 4,
"Ipv6PrefixLen": 56,
"Ipv4BgpNum": 50,
"Ipv4OtherNum": 4,
"RequestId": "aac03e7b-3c91-4970-b2bc-c20f0c6bdd38"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

获取用户互联网公网地址统计信息

最近更新时间：2025-04-25 01:25:36

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

获取用户互联网公网地址分配统计信息

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeInternetAddressStatistics。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。

3. 输出参数

参数名称	类型	描述
TotalCount	Integer	互联网公网地址统计信息数量 示例值：0
InternetAddressStatistics	Array of InternetAddressStatistics	互联网公网地址统计信息列表 示例值：{"Region": "bj", "SubnetNum": 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 获取用户互联网公网地址分配统计信息

输入示例

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

输出示例

```
{
"Response": {
"TotalCount": 1,
"InternetAddressStatistics": [
{
"Region": "gz",
"SubnetNum": 1
}
],
"RequestId": "aac03e7b-3c91-4970-b2bc-c20f0c6bdd38"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

查询互联网通道路由列表

最近更新时间：2025-04-25 01:25:35

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

查询互联网通道路由列表。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribePublicDirectConnectTunnelRoutes。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。
DirectConnectTunnelId	是	String	专用通道ID。 示例值：dcx-xxx
Filters.N	否	Array of Filter	过滤条件： route-type：路由类型，取值：BGP/STATIC； route-subnet：路由cidr，取值如：192.68.1.0/24。
Offset	否	Integer	偏移量，默认为0。 示例值：0
Limit	否	Integer	返回数量，默认为20，最大值为100。 示例值：100

3. 输出参数

参数名称	类型	描述
Routes	Array of DirectConnectTunnelRoute	互联网通道路由列表。
TotalCount	Integer	路由总数量。 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 查询互联网通道路由列表

查询互联网通道路由列表

输入示例

```
https://dc.tencentcloudapi.com/?Action=DescribePublicDirectConnectTunnelRoutes
&DirectConnectTunnelId=dcx-69p738pu
&<公共请求参数>
```

输出示例

```
{
"Response": {
"Routes": [
{
"RouteId": "ipv4-l4udy3uj",
"DestinationCidrBlock": "49.7.252.80/30",
"RouteType": "STATIC",
"Status": "ENABLE",
"ASPath": [],
"NextHop": "10.60.191.1",
"UpdateTime": "2023-01-13 22:48:50",
"ApplyOnTunnelEnable": true
}
],
"RequestId": "8ae32da8-db96-400f-908e-0de2c89e96ea",
"TotalCount": 1
}
}
```

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	内部错误。
InvalidParameter	参数错误。
ResourceNotFound	资源不存在。
ResourceNotFound.DirectConnectTunnelIdIsNotExist	专用通道不存在。

停用公网互联网地址

最近更新时间：2025-04-25 01:25:35

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

停用用户申请的公网互联网地址

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DisableInternetAddress。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。
InstanceId	是	String	公网互联网地址ID 示例值：ipv4-xxx

3. 输出参数

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

4. 示例

示例1 停用公网互联网地址

输入示例

```
https://dc.tencentcloudapi.com/?Action=DisableInternetAddress
&InstanceId="ipv4-qmda2nqv"
&<公共请求参数>
```

输出示例

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

```
"RequestId": "24a0d7e5-4c13-49be-aa16-94f698fbef3e"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

启用互联网公网地址

最近更新时间：2025-04-25 01:25:35

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

启用已停用的互联网公网地址

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：EnableInternetAddress。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。
InstanceId	是	String	互联网公网地址ID 示例值：ipv4-xxx

3. 输出参数

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

4. 示例

示例1 启用互联网公网地址

输入示例

```
https://dc.tencentcloudapi.com/?Action=EnableInternetAddress
&InstanceId="ipv4-qmda2nqv"
&<公共请求参数>
```

输出示例

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

```
"RequestId": "24a0d7e5-4c13-49be-aa16-94f698fbef3e"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

修改物理专线属性

最近更新时间：2025-04-25 01:25:35

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

修改物理专线的属性。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：ModifyDirectConnectAttribute。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。
DirectConnectId	是	String	物理专线ID。 示例值：dcx-abcdefgh
DirectConnectName	否	String	物理专线名称。 示例值：我的专线01
CircuitCode	否	String	运营商或者服务商为物理专线提供的电路编码。 示例值：ABF_123
Vlan	否	Integer	物理专线调试VLAN。 示例值：100
TencentAddress	否	String	物理专线调试腾讯侧互联 IP。 示例值：172.168.1.1/30
CustomerAddress	否	String	物理专线调试用户侧互联 IP。 示例值：172.168.1.2/30
CustomerName	否	String	物理专线申请者姓名。默认从账户体系获取。 示例值：张三
CustomerContactMail	否	String	物理专线申请者联系邮箱。默认从账户体系获取。 示例值：12345@qq.com
CustomerContactNumber	否	String	物理专线申请者联系号码。默认从账户体系获取。 示例值：18812345678
FaultReportContactPerson	否	String	报障联系人。 示例值：张三

参数名称	必选	类型	描述
FaultReportContactNumber	否	String	报障联系电话。 示例值：18812345678
SignLaw	否	Boolean	物理专线申请者补签用户使用协议。 示例值：true
Bandwidth	否	Integer	物理专线带宽。 示例值：1000

3. 输出参数

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

4. 示例

示例1 修改物理专线属性信息

修改物理专线属性信息

输入示例

```
https://dc.tencentcloudapi.com/?Action=ModifyDirectConnectAttribute
&DirectConnectId=dcx-abcdefgh
&DirectConnectName=我的专线01
&CircuitCode=ABF_123
&Vlan=100
&TencentAddress=172.168.1.1/30
&CustomerAddress=172.168.1.2/30
&CustomerName=张三
&CustomerContactMail=12345@qq.com
&CustomerContactNumber=18812345678
&Bandwidth=1000
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "3c140219-cfe9-470e-b241-907877d6fb03"
}
}
```

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	内部错误。
InvalidParameter	参数错误。
InvalidParameter.DirectConnectIdIsNotUin	物理专线不属于该账号。
InvalidParameterValue	参数取值错误。
ResourceNotFound	资源不存在。
ResourceUnavailable.InsufficientBalance	对不起您的账号已欠费，欠费状态下无法开通产品，请您先行充值。
UnauthorizedOperation	未授权操作。
UnsupportedOperation	操作不支持。

修改专用通道属性

最近更新时间：2025-04-25 01:25:35

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

修改专用通道属性。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：ModifyDirectConnectTunnelAttribute。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。
DirectConnectTunnelId	是	String	专用通道ID。 示例值：dcx-xxxxxxxx
DirectConnectTunnelName	否	String	专用通道名称。 示例值：通道名称
BgpPeer	否	BgpPeer	用户侧BGP，包括Asn，AuthKey。
RouteFilterPrefixes.N	否	Array of RouteFilterPrefix	用户侧网段地址。
TencentAddress	否	String	腾讯侧互联IP。 示例值：192.168.0.1/24
CustomerAddress	否	String	用户侧互联IP。 示例值：192.168.0.3/24
Bandwidth	否	Integer	专用通道带宽值，单位为M。 示例值：100
TencentBackupAddress	否	String	腾讯侧备用互联IP。 示例值：192.168.0.2/24

3. 输出参数

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

4. 示例

示例1 修改路由模式是BGP的专用通道

修改路由模式是BGP的专用通道

输入示例

```
https://dc.tencentcloudapi.com/?Action=ModifyDirectConnectTunnelAttribute
&DirectConnectTunnelId=dcx-abcdefgh
&DirectConnectTunnelName=Test
&Bandwidth=100
&TencentAddress=192.168.1.1/30
&CustomerAddress=192.168.1.2/30
&BgpPeer.Asn=65128
&BgpPeer.AuthKey=abcdefg
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "3c140219-cfe9-470e-b241-907877d6fb03"
}
}
```

示例2 修改路由模式是STATIC的专用通道

修改路由模式是STATIC的专用通道

输入示例

```
https://dc.tencentcloudapi.com/?Action=ModifyDirectConnectTunnelAttribute
&DirectConnectTunnelId=dcx-abcdefgh
&DirectConnectTunnelName=Test
&Bandwidth=100
&TencentAddress=192.168.1.1/30
&CustomerAddress=192.168.1.2/30
&RouteFilterPrefixes.0.Cidr=192.168.0.0/24
&RouteFilterPrefixes.1.Cidr=192.168.1.0/24
&RouteFilterPrefixes.2.Cidr=192.168.2.0/24
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "3c140219-cfe9-470e-b241-907877d6fb03"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation	操作失败。
InternalError	内部错误。
InvalidParameter	参数错误。
MissingParameter	缺少参数错误。
ResourceNotFound.DirectConnectTunnelIdIsNotExist	专用通道不存在。
UnsupportedOperation.StateConfLict	状态冲突。

修改专用通道扩展信息

最近更新时间：2025-04-25 01:25:35

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

修改专用通道扩展信息。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：ModifyDirectConnectTunnelExtra。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。
DirectConnectTunnelId	是	String	专用通道ID。 示例值：dcx-r3sml04o
Vlan	否	Integer	专用通道的Vlan。 示例值：90
BgpPeer	否	BgpPeer	Bgp参数，包括Asn，AuthKey
RouteFilterPrefixes	否	RouteFilterPrefix	用户侧过滤网段地址。
TencentAddress	否	String	腾讯侧互联IP。 示例值：192.168.1.1/29
TencentBackupAddress	否	String	腾讯侧备用互联IP。 示例值：192.168.1.1/29
CustomerAddress	否	String	用户侧互联IP。 示例值：192.168.1.1/29
Bandwidth	否	Integer	专用通道带宽值。 示例值：100
EnableBGPCommunity	否	Boolean	BGP community开关。 示例值：false
BfdEnable	否	Integer	是否开启BFD。 示例值：1
NqaEnable	否	Integer	是否开启NQA。 示例值：1

参数名称	必选	类型	描述
BfdInfo	否	BFDInfo	BFD配置信息。
NqaInfo	否	NQAInfo	NQA配置信息。
IPv6Enable	否	Integer	IPV6使能。0：停用IPv6；1: 启用IPv6。 示例值：1
CustomerIDCRoutes.N	否	Array of RouteFilterPrefix	去往用户侧的路由信息。
JumboEnable	否	Integer	是否开启巨帧。1：开启；0：不开启。 示例值：1
TencentIPv6Address	否	String	腾讯侧互联IPv6。 示例值：20::1/64
TencentBackupIPv6Address	否	String	腾讯侧备用互联IPv6。 示例值：20::2/64
CustomerIPv6Address	否	String	用户侧互联IPv6。 示例值：20::3/64

3. 输出参数

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

4. 示例

示例1 修改专用通道扩展信息

修改专用通道扩展信息。

输入示例

```
https://dc.tencentcloudapi.com/?Action=ModifyDirectConnectTunnelExtra
&DirectConnectTunnelId=dcx-r3sml04o
&Vlan=90
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "8ae32da8-db96-400f-908e-0de2c89e96ea"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
FailedOperation	操作失败。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
ResourceInUse	资源被占用。
ResourceNotFound	资源不存在。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

拒绝专用通道申请

最近更新时间：2025-04-25 01:25:35

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

拒绝专用通道申请。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：RejectDirectConnectTunnel。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。
DirectConnectTunnelId	是	String	专用通道ID。 示例值：dcx-xxxxxxxx

3. 输出参数

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

4. 示例

示例1 拒绝专用通道申请

拒绝专用通道申请。

输入示例

```
https://dc.tencentcloudapi.com/?Action=RejectDirectConnectTunnel
&DirectConnectTunnelId=dcx-abcdefgh
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "3c140219-cfe9-470e-b241-907877d6fb03"
}
}
```

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	内部错误。
InvalidParameter	参数错误。
ResourceNotFound.DirectConnectTunnelIdIsNotExist	专用通道不存在。
UnauthorizedOperation	未授权操作。
UnsupportedOperation.StateConfLict	状态冲突。

释放互联网地址

最近更新时间：2025-04-25 01:25:35

1. 接口描述

接口请求域名： dc.tencentcloudapi.com 。

释放已申请的互联网地址

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：ReleaseInternetAddress。
Version	是	String	公共参数，本接口取值：2018-04-10。
Region	否	String	公共参数，此参数为可选参数。
InstanceId	是	String	公网互联网地址ID 示例值：ipv4-xxx

3. 输出参数

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

4. 示例

示例1 释放公网互联网地址

输入示例

```
https://dc.tencentcloudapi.com/?Action=ReleaseInternetAddress
&InstanceId="ipv4-qmda2nqv"
&<公共请求参数>
```

输出示例

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

```
"RequestId": "24a0d7e5-4c13-49be-aa16-94f698fbef3e"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

数据结构

最近更新时间：2025-04-29 01:27:20

AccessPoint

接入点信息。

被如下接口引用：DescribeAccessPoints。

名称	类型	描述
AccessPointName	String	接入点的名称。 示例值：成都-A-西区
AccessPointId	String	接入点唯一ID。 示例值：ap-chengdu-a-xq
State	String	接入点的状态。可用，不可用。 示例值：AVAILABLE
Location	String	接入点的位置。 示例值：成都电信西区803DC
LineOperator	Array of String	接入点支持的运营商列表。 示例值：["ChinaTelecom"]
RegionId	String	接入点管理的大区ID。 示例值：cd
AvailablePortType	Array of String	接入点可用的端口类型列表。1000BASE-T代表千兆电口，1000BASE-LX代表千兆单模光口10km，1000BASE-ZX代表千兆单模光口80km,10GBASE-LR代表万兆单模光口10km,10GBASE-ZR代表万兆单模光口80km,10GBASE-LH代表万兆单模光口40km,100GBASE-LR4代表100G单模光口10km。 示例值：["1000BASE-T", "1000BASE-LX"]
Coordinate	Coordinate	接入点经纬度。 示例值：{"Lat": "0.0", "Lng": "0.0"}
City	String	接入点所在城市。 示例值：成都
Area	String	接入点地域名称。 示例值：华北
AccessPointType	String	接入点类型。VXLAN/QCPL/QCAR 示例值：VXLAN
AvailablePortInfo	Array of PortSpecification	端口规格信息。
Address	String	接入点地址。 示例值：广东省清远市清城区清晖南路腾讯清城云计算数据中心

BFDInfo

BFD配置信息

被如下接口引用：CreateDirectConnectTunnel, DescribeDirectConnectTunnelExtra, ModifyDirectConnectTunnelExtra。

名称	类型	必选	描述
ProbeFailedTimes	Integer	否	健康检查次数 示例值：3
Interval	Integer	否	健康检查间隔 示例值：1000

BGPStatus

bgp状态信息

被如下接口引用：DescribeDirectConnectTunnelExtra。

名称	类型	描述
TencentAddressBgpState	String	腾讯侧主互联IP BGP状态 示例值：Established
TencentBackupAddressBgpState	String	腾讯侧备互联IP BGP状态 示例值：Established

BgpPeer

bgp参数，包括CloudAsn，Asn，AuthKey

被如下接口引用：CreateDirectConnectTunnel, DescribeDirectConnectTunnelExtra, DescribeDirectConnectTunnels, ModifyDirectConnectTunnelAttribute, ModifyDirectConnectTunnelExtra。

名称	类型	必选	描述
CloudAsn	Integer	否	腾讯侧BGP ASN 示例值：45090
Asn	Integer	否	用户侧BGP ASN 示例值：45666
AuthKey	String	否	用户侧BGP密钥 示例值：tencent

CloudAttachInfo

敏捷上云服务信息

被如下接口引用：CreateCloudAttachService。

名称	类型	描述
InstanceId	String	敏捷上云实例id 示例值：cas-xxxxxxxx
Name	String	敏捷上云名称 示例值：名称
IapId	String	合作伙伴的AppId 示例值：1234353
IdcAddress	String	需要接入敏捷上云的IDC的地址 示例值：中国

名称	类型	描述
IdcType	String	需要接入敏捷上云的IDC的互联网服务提供商类型 示例值：CMCC
Bandwidth	Integer	敏捷上云的带宽，单位为MB 示例值：1000
Telephone	String	联系电话 示例值：18600530023
Status	String	敏捷上云的状态 available：就绪状态 applying：申请，待审核状态 pendingpay：代付款状态 building：建设中状态 confirming：待确认状态 isolate: 隔离状态 stoped：终止状态 示例值：pendingpay
ApplyTime	Timestamp	敏捷上云申请的时间 示例值：2020-01-12 12:12:23
ReadyTime	Timestamp	敏捷上云建设完成的时间 示例值：2020-01-12 12:12:23
ExpireTime	Timestamp	敏捷上云过期时间 示例值：2020-01-12 12:12:23
Remarks	String	备注信息 示例值：备注
RegionStatus	String	敏捷上云的地域状态。 same-region：同地域 cross-region：跨地域 示例值：cross-region
AppId	String	用户的AppId 示例值：1235464
Uin	String	用户的Uin 示例值：1234544
CustomerAuthName	String	用户注册名称 示例值：用户注册名称
DirectConnectId	String	物理专线实例ID 示例值：dc-xxxxxxxx
CloudAttachServiceGatewaysSupport	Boolean	敏捷上云是否支持创建高速上云专线网关 示例值：true
BUpdateBandwidth	Boolean	敏捷上云服务是否处于升降配中 示例值：true
ArRegion	String	接入地域 示例值：gz

Coordinate

坐标，经维度描述

被如下接口引用：DescribeAccessPoints。

名称	类型	描述
Lat	Float	纬度 示例值：80.0
Lng	Float	经度 示例值：108.0

CreateCasInput

创建敏捷上云入参

被如下接口引用：CreateCloudAttachService。

名称	类型	必选	描述
Name	String	是	敏捷上云名称 示例值：测试线路名
IdcAddress	String	是	需要接入敏捷上云的IDC的地址 示例值：湖北省武汉市高薪大道xx机房xxx地址
IdcType	String	是	需要接入敏捷上云的IDC的互联网服务提供商类型 示例值：CUCC
Bandwidth	Integer	是	敏捷上云的带宽，单位为MB 示例值：100
Telephone	String	是	联系电话 示例值：13888888888
Remarks	String	是	备注信息 示例值：购买时长6个月
ArRegion	String	否	接入地域 示例值：gz

DirectConnect

物理专线信息列表

被如下接口引用：DescribeDirectConnects。

名称	类型	描述
DirectConnectId	String	物理专线ID。 示例值：dc-psjiehij
DirectConnectName	String	物理专线的名称。 示例值：我的名称01
AccessPointId	String	物理专线的接入点ID。 示例值：ap-beijing-c-jxq
State	String	物理专线的状态。 申请中：PENDING 申请驳回：REJECTED 待付款：TOPAY 已付款：PAID

名称	类型	描述
		建设中：ALLOCATED 已开通：AVAILABLE 删除中：DELETING 已删除：DELETED 。 示例值：PENDINGPAY
CreatedTime	Timestamp ISO8601	物理专线创建时间。 示例值：2020-12-08 16:57:43
EnabledTime	Timestamp ISO8601	物理专线的开通时间。 示例值：2021-01-06 10:39:41
LineOperator	String	提供接入物理专线的运营商。ChinaTelecom：中国电信，ChinaMobile：中国移动，ChinaUnicom：中国联通， In-houseWiring：楼内线，ChinaOther：中国其他，InternationalOperator：境外其他。 示例值：In-houseWiring
Location	String	本地数据中心的地理位置。 示例值：武汉市光谷大道001号机房
Bandwidth	Integer	物理专线接入接口带宽，单位为Mbps。 示例值：1000
PortType	String	用户侧物理专线接入端口类型,取值：100Base-T：百兆电口,1000Base-T（默认值）：千兆电口,1000Base-LX：千兆单模光口（10千米）,10GBase-T：万兆电口10GBase-LR：万兆单模光口（10千米），默认值，千兆单模光口（10千米） 示例值：1000Base-LX
CircuitCode	String	运营商或者服务商为物理专线提供的电路编码。 示例值：abcd1212
RedundantDirectConnectId	String	冗余物理专线的ID。 示例值：dc-1234
Vlan	Integer	物理专线调试VLAN。默认开启VLAN，自动分配VLAN。 示例值：100
TencentAddress	String	物理专线调试腾讯侧互联IP。 示例值：192.168.1.2
CustomerAddress	String	物理专线调试用户侧互联IP。 示例值：192.168.1.1
CustomerName	String	物理专线申请者姓名。默认从账户体系获取。 示例值：张三
CustomerContactMail	String	物理专线申请者联系邮箱。默认从账户体系获取。 示例值：274907341@qq.com
CustomerContactNumber	String	物理专线申请者联系号码。默认从账户体系获取。 示例值：15802738654
ExpiredTime	Timestamp ISO8601	物理专线的过期时间。 示例值：2040-01-06 10:39:41
ChargeType	String	物理专线计费类型。NON_RECURRING_CHARGE：一次性接入费用；PREPAID_BY_YEAR：按年预付费。 示例值：NON_RECURRING_CHARGE

名称	类型	描述
FaultReportContactPerson	String	报障联系人。 示例值：李四
FaultReportContactNumber	String	报障联系电话。 示例值：15802738654
TagSet	Array of Tag	标签键值对 示例值：[{"Key": "abab", "Value": "xxyy"}]
AccessPointType	String	物理专线的接入点类型。 示例值：VXLAN
IdcCity	String	IDC所在城市 示例值：成都
ChargeState	String	计费状态 示例值：NORMAL
StartTime	String	物理专线开通时间 示例值：2020-12-08 16:57:43
SignLaw	Boolean	物理专线是否已签署用户协议 示例值：true
LocalZone	Boolean	物理专线是否为LocalZone 示例值：false
VlanZeroDirectConnectTunnelCount	Integer	该物理专线下vlan 0的专用通道数量 示例值：0
OtherVlanDirectConnectTunnelCount	Integer	该物理专线下非vlan 0的专用通道数量 示例值：0
MinBandwidth	Integer	物理专线最小带宽 示例值：1000
Construct	Integer	建设模式 示例值：1
AccessPointName	String	物理专线的接入点名称 示例值：广州-A-开源路
IsThreeArch	Boolean	是否三层架构 示例值：true

DirectConnectTunnel

专用通道信息列表

被如下接口引用：DescribeDirectConnectTunnels。

名称	类型	描述
DirectConnectTunnelId	String	专用通道ID 示例值：dcx-r3sml04o
DirectConnectId	String	物理专线ID 示例值：dc-9s5kpgyp

名称	类型	描述
State	String	专用通道状态 AVAILABLE:就绪或者已连接 PENDING:申请中 ALLOCATING:配置中 ALLOCATED:配置完成 ALTERING:修改中 DELETING:删除中 DELETED:删除完成 COMFIRMING:待接受 REJECTED:拒绝 示例值：PENDING
DirectConnectOwnerAccount	String	物理专线的拥有者，开发商账号 ID 示例值：2407912486
OwnerAccount	String	专用通道的拥有者，开发商账号 ID 示例值：2407912486
NetworkType	String	网络类型，分别为VPC、BMVPC、CCN VPC：私有网络 ，BMVPC：黑石网络，CCN：云联网 示例值：VPC
NetworkRegion	String	VPC地域对应的网络名，如ap-guangzhou 示例值：ap-guangzhou
VpcId	String	私有网络统一 ID 或者黑石网络统一 ID 示例值：vpc-aipqhdez
DirectConnectGatewayId	String	专线网关 ID 示例值：dcg-r70hz833
RouteType	String	BGP ：BGP路由 STATIC：静态 默认为 BGP 路由 示例值：STATIC
BgpPeer	BgpPeer	用户侧BGP，包括： CloudAsn，Asn，AuthKey 示例值：{"CloudAsn": 45090, "Asn": 16550, "AuthKey": "tencent"}
RouteFilterPrefixes	Array of RouteFilterPrefix	用户侧网段地址 示例值：[{"Cidr": "8.8.8.8/32"}]
Vlan	Integer	专用通道的Vlan 示例值：1001
TencentAddress	String	TencentAddress，腾讯侧互联 IP 示例值：169.254.64.1/29
CustomerAddress	String	CustomerAddress，用户侧互联 IP 示例值：169.254.64.2/29
DirectConnectTunnelName	String	专用通道名称 示例值：测试通道
CreatedTime	Timestamp ISO8601	专用通道创建时间 示例值：2018-06-01 14:59:16
Bandwidth	Integer	专用通道带宽值 示例值：1000
TagSet	Array of Tag	专用通道标签值 示例值：[{"Key": "xxyy", "Value": "aabb"}]

名称	类型	描述
NetDetectId	String	关联的网络自定义探测ID 示例值：netd-aabb1212
EnableBGPCommunity	Boolean	BGP community开关 示例值：False
NatType	Integer	是否为Nat通道 示例值：1
VpcRegion	String	VPC地域简码，如gz、cd 示例值：gz
BfdEnable	Integer	是否开启BFD 示例值：1
AccessPointType	String	专用通道接入点类型 示例值：QCPL
DirectConnectGatewayName	String	专线网关名称 示例值：测试网关
VpcName	String	VPC名称 示例值：测试VPC
TencentBackupAddress	String	TencentBackupAddress，腾讯侧备用互联 IP 示例值：169.254.64.3/29
SignLaw	Boolean	专用通道关联的物理专线是否签署了用户协议 示例值：True
CloudAttachId	String	高速上云服务ID 示例值：cas-xxxxxxxx
ShareOrNot	Integer	是否共享通道 示例值：1

DirectConnectTunnelExtra

专用通道扩展信息

被如下接口引用：DescribeDirectConnectTunnelExtra。

名称	类型	描述
DirectConnectTunnelId	String	专用通道ID 示例值：dcx-xxxxxxxx
DirectConnectId	String	物理专线ID 示例值：dc-xxxxxxxx
State	String	专用通道状态 AVAILABLE:就绪或者已连接 PENDING:申请中 ALLOCATING:配置中 ALLOCATED:配置完成 ALTERING:修改中 DELETING:删除中 DELETED:删除完成 COMFIRMING:待接受

名称	类型	描述
		REJECTED:拒绝 示例值：ALLOCATING
DirectConnectOwnerAccount	String	物理专线的拥有者，开发商账号 ID 示例值：1233453
OwnerAccount	String	专用通道的拥有者，开发商账号 ID 示例值：12343252
NetworkType	String	网络类型，分别为VPC、BMVPC、CCN VPC：私有网络 ，BMVPC：黑石网络，CCN：云联网 示例值：VPC
NetworkRegion	String	VPC地域对应的网络名，如ap-guangzhou 示例值：ap-guangzhou
VpcId	String	私有网络统一 ID 或者黑石网络统一 ID 示例值：vpc-xxxxxxxx
DirectConnectGatewayId	String	专线网关 ID 示例值：dcx-xxxxxxxx
RouteType	String	BGP：BGP路由 STATIC：静态 默认为 BGP 路由 示例值：STATIC
BgpPeer	BgpPeer	用户侧BGP，Asn，AuthKey 示例值：{"CloudAsn": 45090, "Asn": 16550, "AuthKey": "tencent"}
RouteFilterPrefixes	Array of RouteFilterPrefix	用户侧网段地址 示例值：[{"Cidr": "8.8.8.8/32"}]
PublicAddresses	Array of RouteFilterPrefix	互联网通道公网网段地址 示例值：[{"Cidr": "9.9.9.9/32"}]
Vlan	Integer	专用通道的Vlan 示例值：100
TencentAddress	String	腾讯侧互联 IP 示例值：192.168.1.1/29
TencentBackupAddress	String	腾讯侧备用互联IP 示例值：192.168.1.2/29
CustomerAddress	String	用户侧互联 IP 示例值：192.168.1.3/29
DirectConnectTunnelName	String	专用通道名称 示例值：通道
CreatedTime	Timestamp ISO8601	专用通道创建时间 示例值：2020-09-23 00:00:00
Bandwidth	Integer	专用通道带宽值 示例值：2020-09-23 00:00:00
NetDetectId	String	关联的网络自定义探测ID 示例值：netid-xxxxxxxx
EnableBGPCommunity	Boolean	BGP community开关 示例值：true

名称	类型	描述
NatType	Integer	是否为Nat通道 示例值：1
VpcRegion	String	VPC地域简码，如gz、cd 示例值：gz
BfdEnable	Integer	是否开启BFD 示例值：1
NqaEnable	Integer	是否开启NQA 示例值：1
AccessPointType	String	专用通道接入点类型 示例值：1
DirectConnectGatewayName	String	专线网关名称 示例值：网关
VpcName	String	VPC名称 示例值：VPC
SignLaw	Boolean	专用通道关联的物理专线是否签署了用户协议 示例值：true
BfdInfo	BFDInfo	BFD配置信息 示例值：{"ProbeFailedTimes": 3, "Interval": 3000}
NqaInfo	NQAInfo	NQA配置信息 示例值：{"ProbeFailedTimes": 3, "Interval": 3000, "DestinationIp": "8.8.8.8"}
BgpStatus	BGPStatus	BGP状态 示例值：{"TencentAddressBgpState": "Established", "TencentBackupAddressBgpState": "Established"}
IPv6Enable	Integer	是否开启IPv6 示例值：1
TencentIPv6Address	String	腾讯侧互联IPv6地址 示例值：2402:4e00:4011:1004::5/126
TencentBackupIPv6Address	String	腾讯侧备用互联IPv6地址 示例值：2402:4e00:4011:1004::5/126
BgpIPv6Status	BGPStatus	BGPv6状态 示例值：{"TencentAddressBgpState": "Established", "TencentBackupAddressBgpState": "Established"}
CustomerIPv6Address	String	用户侧互联IPv6地址 示例值：2402:4e00:4011:1004::5/126
JumboEnable	Integer	专用通道是否支持巨帧。1 支持，0 不支持 示例值：1
HighPrecisionBFDEnable	Integer	专用通道是否支持高精度BFD。1支持，0不支持 示例值：1

DirectConnectTunnelRoute

专用通道路由

被如下接口引用：DescribePublicDirectConnectTunnelRoutes。

名称	类型	描述
RouteId	String	专用通道路由ID 示例值：dcxr-69sg663r
DestinationCidrBlock	String	网段CIDR 示例值：10.0.0.0/16
RouteType	String	路由类型：BGP/STATIC路由 示例值：BGP
Status	String	ENABLE：路由启用，DISABLE：路由禁用 示例值：ENABLE
ASPath	Array of String	ASPath信息 示例值：['123', '342']
NextHop	String	路由下一跳IP 示例值：0.0.0.0
UpdateTime	String	路由更新时间 示例值：2021-07-11 12:32:32
ApplyOnTunnelEnable	Boolean	是否配置在通道上 示例值：True

Filter

用于条件过滤查询

被如下接口引用：DescribeAccessPoints, DescribeDirectConnectTunnels, DescribeDirectConnects, DescribeInternetAddress, DescribePublicDirectConnectTunnelRoutes。

名称	类型	必选	描述
Name	String	是	需要过滤的字段。 示例值：Version
Values	Array of String	是	字段的过滤值。 示例值：V3

InternetAddressDetail

互联网地址详细信息

被如下接口引用：DescribeInternetAddress。

名称	类型	描述
InstanceId	String	互联网地址ID 示例值：ipv4-qmda2nqq
Subnet	String	互联网网络地址 示例值：220.110.1.0
MaskLen	Integer	网络地址掩码长度 示例值：30

名称	类型	描述
AddrType	Integer	0:BGP 1:电信 2:移动 3:联通 示例值：0
Status	Integer	0:使用中 1:已停用 2:已退还 示例值：0
ApplyTime	String	申请时间 示例值：2020-09-22 00:00:00
StopTime	String	停用时间 示例值：2020-09-22 00:00:00
ReleaseTime	String	退还时间 示例值：2020-09-22 00:00:00
Region	String	地域信息 示例值：gz
AppId	Integer	用户ID 示例值：251010426
AddrProto	Integer	0:IPv4 1:IPv6 示例值：0
ReserveTime	Integer	释放状态的IP地址保留的天数 示例值：2020-09-22 00:00:00

InternetAddressStatistics

互联网公网地址统计

被如下接口引用：DescribeInternetAddressStatistics。

名称	类型	描述
Region	String	地域 示例值：gz
SubnetNum	Integer	互联网公网地址数量 示例值：64

NQAInfo

nqa配置信息

被如下接口引用：CreateDirectConnectTunnel, DescribeDirectConnectTunnelExtra, ModifyDirectConnectTunnelExtra。

名称	类型	必选	描述
ProbeFailedTimes	Integer	否	健康检查次数 示例值：3
Interval	Integer	否	健康检查间隔

名称	类型	必选	描述
			示例值：5
DestinationIp	String	否	健康检查地址 示例值：8.8.8.8

PortSpecification

端口规格

被如下接口引用：DescribeAccessPoints。

名称	类型	描述
InternationalName	String	端口名称 示例值：10GBASE-LR
Specification	Integer	端口规格（M） 示例值：10000
PortType	String	端口类型：T-电口，X-光口 示例值：X

RouteFilterPrefix

用户侧网段地址

被如下接口引用：CreateDirectConnectTunnel, DescribeDirectConnectTunnelExtra, DescribeDirectConnectTunnels, ModifyDirectConnectTunnelAttribute, ModifyDirectConnectTunnelExtra。

名称	类型	必选	描述
Cidr	String	否	用户侧网段地址 示例值：8.8.8.8/32

Tag

标签键值对

被如下接口引用：CreateDirectConnect, CreateDirectConnectTunnel, DescribeDirectConnectTunnels, DescribeDirectConnects。

名称	类型	必选	描述
Key	String	是	标签键 示例值：xxyy
Value	String	是	标签值 示例值：abab

错误码

最近更新时间：2024-12-20 01:23:04

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
AuthFailure	CAM签名/鉴权错误。
InvalidParameter.AddressError	互联IP错误。
InvalidParameter.DirectConnectIdIsNotUin	物理专线不属于该账号。
InvalidParameter.UinIsNotExist	该账号ID不存在。
InvalidParameter.VlanConflict	vlan冲突。
InvalidParameterValue.VlanConfLict	vlan冲突。

错误码	说明
LimitExceeded.DirectConnectLimitExceeded	物理专线数已达上限。
LimitExceeded.DirectConnectTunnelLimitExceeded	物理专线的专用通道数已达上限。
ResourceInUse.DcVpcIsExist	物理专线的vpc已经存在。
ResourceNotFound.DirectConnectTunnelIdIsNotExist	专用通道不存在。
ResourceUnavailable.InsufficientBalance	对不起您的账号已欠费，欠费状态下无法开通产品，请您先行充值。
UnsupportedOperation.CrossBorderDirectConnectTunnel	不允许创建跨境专用通道，请您联系我们。
UnsupportedOperation.StateConfLict	状态冲突。

专线接入 API 2017
简介

最近更新时间：2020-06-04 10:07:36

说明:

- 当前页面接口为旧版 API，未来可能停止维护，目前不展示在左侧导航。专线接入 API 3.0 版本接口定义更加规范，访问时延下降显著，建议使用 专线接入 API 3.0 。
- 如果您需要访问专线接入旧版 API，可参见 API 概览 。

欢迎使用腾讯云专线接入(DC)。本文档提供的 API 供您使用请求调用的方式来操作专线接入，具体支持的操作可参见 API 概览 。请确保在使用这些接口前，已充分了解 DC 产品、及其使用和收费方式。
在本文档的接口说明部分，凡出现任何参数可选范围等方面与腾讯云官网上给出的数值发生矛盾时，**均以官网上给出的值为准**。

API概览

最近更新时间：2019-08-09 16:10:21

接口功能	Action ID
查询物理专线列表	DescribeDirectConnects
创建专用通道	CreateDirectConnectTunnel
查询专用通道列表	DescribeDirectConnectTunnels

调用方式
请求结构

最近更新时间：2019-08-09 16:11:07

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

服务地址

腾讯云 API 的服务接入地址与具体模块相关，详细请参见各接口相关描述。

通信协议

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

请求方法

腾讯云 API 同时支持 POST 和 GET 请求。

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

请求参数

腾讯云 API 的每个请求都需要指定两类参数：公共请求参数以及接口请求参数。其中公共请求参数是每个接口都要用到的请求参数，具体可参见 公共请求参数，而接口请求参数是各个接口所特有的，具体见各个接口的“请求参数”描述。

字符编码

腾讯云 API 的请求及返回结果均使用 UTF-8 字符集进行编码。

接口鉴权

最近更新时间：2024-11-08 14:28:04

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

申请安全凭证

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

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

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

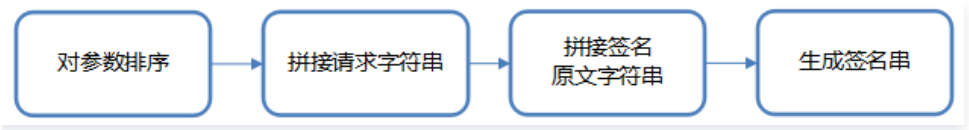
申请安全凭证步骤：

1. 登录腾讯云控制台，进入 API 密钥管理 页面。
2. 在 API 密钥管理页面，单击【新建密钥】即可以创建一对 SecretId/SecretKey。

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

生成签名串

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



假设用户的 SecretId 和 SecretKey 分别是：
SecretId：`**********`
SecretKey：`Gu5t****pq86cd98joQYCN3Cozk1qA`

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

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

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

Region	实例所在区域	ap-guangzhou
SignatureMethod	签名方式	HmacSHA256
InstanceIds.0	待查询的实例 ID	ins-09dx96dg

1. 对参数排序

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

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

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

2. 拼接请求字符串

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

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

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

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

3. 拼接签名原文字符串

此步骤将生成签名原文字符串。
签名原文字符串的拼接规则为：

> **请求方法 + 请求主机 +请求路径 + ? + 请求字符串**

参数构成说明：

- **请求方法：** 支持 POST 和 GET 方式，这里使用 GET 请求， 注意方法为全大写。
- **请求主机：** 即主机域名，请求域名由接口所属的产品或所属产品的模块决定，不同产品或不同产品的模块的请求域名会有不同。如腾讯云 CVM 的查询实例列表（DescribeInstances）的请求域名为：`cvm.api.qcloud.com`，具体产品请求域名详见各接口说明。

- **请求路径：** 腾讯云 API 对应产品的请求路径，一般是一个产品对应一个固定路径，如腾讯云 CVM 请求路径固定为 `/v2/index.php` 。
- **请求字符串：** 即上一步生成的请求字符串。

因此，上述示例的拼接签名原文字符串结果为（请忽略文中的换行）：

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

4. 生成签名串

此步骤生成签名串。

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

首先使用签名算法（HmacSHA256 或 HmacSHA1）对上一步中获得的 **签名原文字符串** 进行签名，然后将生成的签名串使用 Base64 进行编码，即可获得最终的签名串。
具体代码如下，以 PHP 语言为例，由于本例中所用的签名算法为 **HmacSHA256**，因此生成签名串的代码如下（使用其它程序设计语言开发时，可用上述示例中的原文字符串进行签名验证，得到的签名串与例子中的一致即可）：

```
$secretKey = 'Gu5t****pq86cd98joQYCN3Cozk1qA';
$srcStr = 'GETcvm.api.qcloud.com/v2/index.php?Action=DescribeInstances&InstanceIds.0=ins-
09dx96dg&Nonce=11886&Region=ap-
guangzhou&SecretId=***********************************&SignatureMethod=HmacSHA256&Timestamp=1465185
768';
$signStr = base64_encode(hash_hmac('sha256', $srcStr, $secretKey, true));
echo $signStr;
```

最终得到的签名串为：

```
0EEm/HtGRr/VJXTAD9tYMth1Bzm3lLHz5RCDv1GdM8s=
```

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

```
$secretKey = 'Gu5t9xGARNpq86cd98joQYCN3Cozk1qA';
$srcStr = 'GETcvm.api.qcloud.com/v2/index.php?Action=DescribeInstances&InstanceIds.0=ins-
09dx96dg&Nonce=11886&Region=ap-
guangzhou&SecretId=***********************************&SignatureMethod=HmacSHA1&Timestamp=146518576
8';
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));
echo $signStr;
```

最终得到的签名串为：

```
nPVnY6njQmwQ8ciqbPl5Qe+Oru4=
```

签名串编码

生成的签名串并不能直接作为请求参数，需要对其进行 URL 编码。
如上一步生成的签名串为 `0EEm/HtGRr/VJXTAD9tYMth1Bzm3lLHz5RCDv1GdM8s=` ，则其编码后为 `0EEm%2FHtGRr%2FVJXTAD9tYMth1Bzm3lLHz5RCDv1GdM8s%3D` 。因此，最终得到的签名串请求参数 (Signature) 为：`0EEm%2FHtGRr%2FVJXTAD9tYMth1Bzm3lLHz5RCDv1GdM8s%3D` ，它将用于生成最终的请求URL。

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

鉴权失败

当鉴权不通过时，可能出现如下表的错误：

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

公共参数

最近更新时间：2020-01-14 10:13:16

一个完整的腾讯云 API 请求需要两类请求参数：公共请求参数和接口请求参数。本文将介绍腾讯云 API 请求需要用到的6个公共请求参数，有关接口请求参数的详细说明请参见 接口请求参数 章节。
公共请求参数是每个接口都需要使用到的请求参数，开发者每次使用腾讯云 API 发送请求时都需要携带这些公共请求参数，否则会导致请求失败。公共请求参数的首字母均为大写，以此区分于接口请求参数。
公共请求参数具体列表如下：

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

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

使用示例

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

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

返回值
返回值结构

最近更新时间：2020-03-11 15:33:55

如无特别说明，每次请求的返回值中，都会包含下面的字段：

名称	类型	描述
code	Int	返回结果的错误码，0表示成功，其它值表示失败。具体错误码的含义可以参考 错误码 页面
message	String	请求结果

例如：

使用公共参数部分的示例请求：

```
https://domain/v2/index.php?Action=DescribeInstances&SecretId=xxxxxxx&Region=gz
&Timestamp=1402992826&Nonce=345122&Signature=mysignature&instanceId=101
```

可能的返回结果如下：

```
{
    "code":0,
    "message": "success",
    "instanceSet":
   [{
        "instanceId":"qcvm1234",
        "cpu":1,
        "mem":2,
        "disk":20,
        "bandwidth":65535,
        "os":"centos_62_64",
        "lanIp":"10.207.248.186",
        "wanIp":null,
        "status":0
   }]
}
```

错误码

最近更新时间：2023-08-09 09:43:28

响应包体内的返回码(code), 反映了腾讯云API调用和执行的概要结果。
当返回码不为 0 时, 表示请求未正常执行, 返回码也称为错误码, 错误描述(message)对该结果进行了细化补充，用户可根据错误码判断API的执行情况。
message在部分终端（例如浏览器），中文会显示成 unicode 编码，需要解码。
腾讯云 API 可能返回的错误码表如下：

错误代码	错误类型	描述
4000	请求参数非法	缺少必要参数，或者参数值格式不正确，具体错误信息请查看错误描述 message 字段。
4100	鉴权失败	签名鉴权失败，请参考文档中鉴权部分。
4200	请求过期	请求已经过期，请参考文档中请求有效期部分。
4300	拒绝访问	账号被封禁，或者不在接口针对的用户范围内等。
4400	超过配额	请求的次数超过了配额限制，请参考文档请求配额部分。
4500	重放攻击	请求的 Nonce 和 Timestamp 参数用于确保每次请求只会在服务器端被执行一次, 所以本次的 Nonce 和上次的不能重复, Timestamp 与腾讯服务器相差不能超过 2 小时。
4600	协议不支持	协议不支持，请参考文档说明。
5000	资源不存在	资源标识对应的实例不存在，或者实例已经被退还，或者访问了其他用户的资源。
5100	资源操作失败	对资源的操作失败，具体错误信息请查看错误描述 message 字段，稍后重试或者联系客服人员帮忙解决。
5200	资源购买失败	购买资源失败，可能是不支持实例配置，资源不足等等。
5300	资源购买失败	购买资源失败，余额不足。
5400	部分执行成功	批量操作部分执行成功, 详情见方法返回值。
5500	用户资质审核未通过	购买资源失败，用户资质审核未通过。
6000	服务器内部错误	服务器内部出现错误，请稍后重试或者联系客服人员帮忙解决。
6100	版本暂不支持	本版本内不支持此接口或该接口处于维护状态等。注意: 出现这个错误时, 请先确定接口的域名是否正确, 不同的模块, 域名可能不一样。
6200	接口暂时无法访问	当前接口处于停服维护状态，请稍后重试。

物理专线
查询专线列表

最近更新时间：2019-08-09 16:17:59

功能描述

DescribeDirectConnects 用于查询专线列表。
接口请求域名： `dc.api.qcloud.com`

请求

语法示例：

```
GET https://dc.api.qcloud.com/v2/index.php?Action=DescribeDirectConnects
    &<公共请求参数>
    &directConnectId=dc-kd7d06of
```

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，见 公共请求参数 页面。其中，此接口的 Action 字段为 DescribeDirectConnects。

参数名称	是否必选	类型	描述
directConnectId	否	String	专线 ID，例如：dc-kd7d06of，不传返回开发商创建的所有专线。

响应

响应示例：

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

响应参数

参数名称	类型	描述
code	Int	错误码，0: 成功，其他值: 失败。
message	String	错误信息。
data.n	Array	返回的数组。
data.n.directConnectId	String	系统分配的专线ID，例如：dc-kd7d06of。
data.n.directConnectName	String	专线名称。

data.n.status	String	专线状态，0：运行中；1：已到期；2：删除中；3：已删除；11：申请中；12：申请驳回；13：待付款；14：已付款；15：建设中；16：施工停止；
data.n.provider	String	专线运营商。
data.n.portType	int	接口类型。1:100Base-T百兆电口；2: 1000Base-T 千兆电口；3: 1000Base-LX 千兆单模光口（10km）；4: 10GBase-LR 万兆单模光口（10km）
data.n.accessPoints	String	接入点。
data.n.bandwidth	String	专线带宽，单位Mbps。
data.n.loaIssueTime	String	专线到期时间。

实际案例

请求

```
GET https://dc.api.qcloud.com/v2/index.php?Action=DescribeDirectConnects
    &<<a href="https://cloud.tencent.com/doc/api/229/6976">公共请求参数</a>>
    &directConnectId=dc-kd7d06of
```

响应

```
{
    "code": 0,
    "message": "",
    "data": [
        {
            "directConnectId": "dc-3cavza1z",
            "directConnectName": "李昊测试专线@123112",
            "status": 15,
            "provider": "中国电信",
            "portType": 8,
            "accessPoints": "1",
            "bandwidth": 200,
            "loaIssueTime": "2017-09-02 15:41:00"
        }
    ]
}
```

专用通道
创建专用通道

最近更新时间：2019-08-09 16:25:40

接口描述

用于创建专用通道的接口。
域名：`dc.api.qcloud.com`
接口名：`CreateDirectConnectTunnel`

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。其中，此接口的 Action字 段为 `CreateDirectConnectTunnel` 。

参数	必选	类型	描述
directConnectId	是	String	专线 ID，例如 dc-kd7d06of
directConnectTunnelName	是	String	专用通道名称
ownerAccount	否	String	物理专线 owner，缺省为当前客户（物理专线 owner） 共享专线时这里需要填写共享专线的开发商账号 ID
networkType	否	Int	网络类型，默认为 1 1：私有网络 0：黑石网络
region	是	String	网络地域
vpcId	是	String	私有网络统一 ID 或者黑石网络统一 ID
directConnectGatewayId	是	String	专线网关 ID，例如 dcg-d545ddf
bandwidth	否	Int	专线带宽，单位：Mbps ，默认值是物理专线带宽值，带宽梯度， 2,4,6,8,10,20,30,40,50,60,70,80,90,100,200,300,400,500,600,700,800,900,1000,2000,3000,4000,5000,6000,7000,8000,9000,10000,40000,100000
routeMode	否	Int	0：BGP 路由 1：静态 默认为 BGP 路由
bgpPeers.asn	否	String	BGP asn
bgpPeers.authKey	否	String	BGP 密钥
routeFilterPrefixes.n.cidr	否	String	对端网段
vlanId	是	Int	vlanId，范围：0 - 3000 0：不开启子接口
localGatewayIp	否	String	localGatewayIp，腾讯侧互联 IP
peerGatewayIp	否	String	peerGatewayIp，用户侧互联 IP

peeringSubnetMask	否	String	互联 IP 的掩码，必须定义在同一子网内，支持24 – 30位，点分十进制，如255.255.255.252
remark	否	String	备注

响应参数

参数	类型	描述
code	Int	错误码 0：成功 其他值：失败
message	String	错误信息
directConnectTunnelId	String	专用通道 ID

代码示例

请求示例

```
GET https://dc.api.qcloud.com/v2/index.php?Action=CreateDirectConnectTunnel
&<<a href="https://cloud.tencent.com/doc/api/229/6976">公共请求参数</a>>
&directConnectId=dc-kd7d06of
&directConnectTunnelName=baytest
&region=gz
&vpcId=vpc-abcdefg
&directConnectGatewayId=dcg-abcdefg
&vlanId=400
&routeMode=1
&routeFilterPrefixes.0.cidr=172.256.12.0/24
&routeFilterPrefixes.1.cidr=172.256.13.0/24
&localGatewayIp=169.254.64.1
&peerGatewayIp=169.254.64.2
&peeringSubnetMask=255.255.255.252
&remark=create
```

响应示例

```
{
    "code": 0,
    "message": "",
    "directConnectTunnelId": ""
}
```

修改专用通道

最近更新时间：2019-08-09 16:20:01

功能描述

用于修改专用通道参数的接口。
域名： `dc.api.qcloud.com`
接口名： `ModifyDirectConnectTunnel`

请求参数

参数	类型	必选	描述
directConnectTunnelId	String	是	专用通道 ID 例如 dcx-abcdefgh
directConnectTunnelName	String	否	专用通道名称
peerAsn	Int	否	用户侧，BGP asn
authKey	String	否	用户侧，BGP 密钥
routeFilterPrefixes.n.cidr	String	否	用户侧，对端网段 例如 169.254.0.0/28
localGatewayIp	String	否	localGatewayIp，腾讯侧互联 IP
peerGatewayIp	String	否	peerGatewayIp，用户侧互联 IP
peeringSubnetMask	String	否	互联 IP 的掩码，必须定义在同一子网内，支持24～30位，点分十进制，如255.255.255.252

响应参数

参数	类型	描述
code	Int	错误码 0：成功 其他值：失败
message	String	错误信息

代码示例

请求示例

```
GET https://dc.api.qcloud.com/v2/index.php?Action=DeleteDirectConnectTunnel
&<<a href="https://cloud.tencent.com/doc/api/229/6976">公共请求参数</a>>
&directConnectTunnelId=dcx-abcdefgh
&bandwidth=10
&routeFilterPrefixes.0.cidr=172.256.12.0/24
&routeFilterPrefixes.1.cidr=172.256.13.0/24
&localGatewayIp=169.254.64.1
&peerGatewayIp=169.254.64.2
&peeringSubnetMask=255.255.255.252
```

响应示例

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

删除专用通道

最近更新时间：2019-08-09 16:21:22

功能描述

用于删除专用通道的接口。
域名： dc.api.qcloud.com
接口名： DeleteDirectConnectTunnel

请求参数

参数	类型	必选	描述
directConnectTunnelId	String	是	专用通道 ID 例如 dcx-abcdefgh

响应参数

参数	类型	描述
code	Int	错误码 0：成功 其他值： 失败
message	String	错误信息

代码示例

请求示例

```
GET https://dc.api.qcloud.com/v2/index.php?Action=DeleteDirectConnectTunnel
&<<a href="https://cloud.tencent.com/doc/api/229/6976">公共请求参数</a>>
&directConnectTunnelId=dcx-abcdefgh
```

响应示例

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

查询专用通道列表

最近更新时间：2019-08-09 16:15:57

功能描述

用于查询专用通道列表。
域名：`dc.api.qcloud.com`
接口名：`DescribeDirectConnectTunnels`

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。其中，此接口的 Action 字段为 `DescribeDirectConnectTunnels` 。

参数	必选	类型	描述
directConnectId	否	String	专线 ID 例如 dc-kd7d06of
directConnectTunnelId	否	String	专用通道 ID 例如 dcx-kd7d0125

响应参数

参数	类型	描述
code	Int	错误码 0：成功 其他值：失败
message	String	错误信息
data.n	Array	返回的数组
data.n.directConnectTunnelId	String	系统分配的专用通道 ID 例如 dcx-kd7d0125
data.n.directConnectTunnelName	String	专用通道名称
data.n.directConnectId	String	系统分配的专线 ID 例如 dc-kd7d06of
data.n.ownerAccount	String	专线的开发商账号 ID
data.n.networkType	Int	0：黑石网络 1：私有网络
data.n.region	String	网络地域
data.n.vpcId	String	私有网络统一 ID 或者黑石网络统一 ID
data.n.directConnectGatewayId	String	专线网关 ID 例如 dcg-d545ddf
data.n.bandwidth	Int	专线带宽，单位：Mbps
data.n.routeMode	Int	0：BGP 路由 1：静态 默认为 BGP 路由

data.n.bgpPeers.asn	string	BGP asn
data.n.bgpPeers.authKey	String	BGP 密钥
data.n.routeFilterPrefixes.n.cidr	String	对端网段
data.n.status	Int	专用通道状态 0：已连接 1：申请中 2：配置中 6：配置完成 20：等待连接 21：拒绝
data.n.vlan	Int	vlan Id
data.n.localGatewayIp	String	腾讯侧互联 IP
data.n.peerGatewayIp	String	用户侧互联 IP
data.n.peeringSubnetMask	String	互联 IP 的掩码 例如 255.255.255.252
data.n.remark	String	备注

代码示例

请求示例

```
GET https://dc.api.qcloud.com/v2/index.php?Action=DescribeDirectConnectTunnels
&<<a href="https://cloud.tencent.com/doc/api/229/6976">公共请求参数</a>>
&directConnectId=dc-kd7d06of
```

响应示例

```
{
    "code": 0,
    "message": "",
    "data": [
        {
            "directConnectTunnelId": "dcx-2nakhj58",
            "directConnectTunnelName": "barrytest2",
            "directConnectId": "dc-5e8ak079",
            "networkType": 1,
            "ownerAccount": "",
            "region": "gz",
            "vpcId": "vpc-kx49lmyv",
            "bandwidth": 0,
            "routeMode": 0,
            "bgpPeers": {
                "asn": "10",
                "authKey": "124545d"
            },
            "routeFilterPrefixes": [
                {
                    "cidr": ""
```

```
                }
            ],
            "status": 1,
            "vlan": 0,
            "localGatewayIp":"169.254.64.1",
            "peerGatewayIp":"169.254.64.2",
            "peeringSubnetMask":"255.255.255.252",
            "remark": ""
        }
    ]
}
```

接受专用通道申请

最近更新时间：2019-08-09 16:22:41

功能描述

用于接受专用通道的接口。
域名：`dc.api.qcloud.com`
接口名：`AcceptDirectConnectTunnel`

请求参数

参数	类型	必选	描述
directConnectTunnelId	String	是	专用通道 ID 例如 dcx-abcdefgh

响应参数

参数	类型	描述
code	Int	错误码 0：成功 其他值：失败
message	String	错误信息

代码示例

请求示例

```
GET https://dc.api.qcloud.com/v2/index.php?Action=AcceptDirectConnectTunnel
&<<a href="https://cloud.tencent.com/doc/api/229/6976">公共请求参数</a>>
&directConnectTunnelId=dcx-abcdefgh
```

响应示例

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

拒绝专用通道申请

最近更新时间：2019-08-09 16:23:56

功能描述

用于拒绝专用通道的接口。
域名：`dc.api.qcloud.com`
接口名：`RefuseDirectConnectTunnel`

请求参数

参数	类型	必选	描述
directConnectTunnelId	String	是	专用通道 ID 例如 dcx-abcdefgh

响应参数

参数	类型	描述
code	Int	错误码 0：成功 其他值：失败
message	String	错误信息

代码示例

请求示例

```
GET https://dc.api.qcloud.com/v2/index.php?Action=RefuseDirectConnectTunnel
&<<a href="https://cloud.tencent.com/doc/api/229/6976">公共请求参数</a>>
&directConnectTunnelId=dcx-abcdefgh
```

响应示例

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