

云拨测 API 文档



腾讯云

【版权声明】

©2013-2025 腾讯云版权所有

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

【商标声明】

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

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。

您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

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

文档目录

API 文档

- 更新历史

- 简介

- API 概览

- 调用方式

 - 请求结构

 - 公共参数

 - 签名方法 v3

 - 签名方法

 - 返回结果

 - 参数类型

- 拨测资源（旧版）相关接口

 - 查询拨测节点

- 云拨测（新版）相关接口

 - 批量更新拨测任务配置

 - 暂停任务

 - 恢复拨测任务

 - 列出单次拨测详情数据

 - 删除任务

 - 分页查询拨测任务列表

 - 批量创建拨测任务

 - 列出云拨测指标详细数据

 - 获取拨测节点

 - 更新探测任务属性

 - 获取历史即时拨测任务

- 数据结构

- 错误码

API 文档

更新历史

最近更新时间：2024-11-15 01:13:41

第 19 次发布

发布时间：2024-11-15 01:13:36

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- [UpdateProbeTaskConfigurationList](#)
 - 新增入参：BatchTasks

第 18 次发布

发布时间：2024-07-26 01:15:04

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- [CreateProbeTasks](#)
 - 新增入参：SubSyncFlag

第 17 次发布

发布时间：2024-06-12 01:16:37

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- [ProbeTask](#)
 - 新增成员：SubSyncFlag

第 16 次发布

发布时间：2023-12-26 01:07:57

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- [DescribeDetailedSingleProbeData](#)
 - 新增入参：QueryFlag

第 15 次发布

发布时间：2023-11-24 01:07:54

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [DescribeInstantTasks](#)

修改接口：

- [CreateProbeTasks](#)
 - 新增入参：NodeIpType
- [UpdateProbeTaskConfigurationList](#)
 - 新增入参：NodeIpType

新增数据结构：

- [SingleInstantTask](#)

修改数据结构：

- [ProbeTask](#)
 - 新增成员：NodeIpType

第 14 次发布

发布时间：2023-10-25 01:08:41

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- [DescribeDetailedSingleProbeData](#)
 - 新增入参：ScrollIID
 - 新增出参：ScrollIID

第 13 次发布

发布时间：2022-11-09 06:14:16

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [UpdateProbeTaskAttributes](#)

第 12 次发布

发布时间：2022-08-26 06:12:22

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- [UpdateProbeTaskConfigurationList](#)
 - 新增入参：ResourceIDs

第 11 次发布

发布时间：2022-08-02 06:06:56

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [DescribeNodes](#)

新增数据结构：

- [NodeDefineExt](#)

第 10 次发布

发布时间：2022-06-15 06:04:25

本次发布包含了以下内容：

改善已有的文档。

删除接口：

- BindAlarmPolicy
- CreateAgentGroup
- CreateTaskEx
- DeleteAgentGroup
- DeleteTasks
- DescribeAgentGroups
- DescribeAgents
- DescribeAlarmTopic
- DescribeAlarms
- DescribeAlarmsByTask
- DescribeCatLogs
- DescribeTaskDetail
- DescribeTasksByType
- DescribeUserLimit
- GetAvailRatioHistory
- GetDailyAvailRatio
- GetRealAvailRatio
- GetRespTimeTrendEx
- GetResultSummary
- GetReturnCodeHistory
- GetReturnCodeInfo
- GetTaskTotalNumber
- ModifyAgentGroup
- ModifyTaskEx
- PauseTask
- RunTask
- VerifyResult

删除数据结构：

- AgentGroup
- AlarmInfo
- AlarmTopic
- CatAgent
- CatLog
- CatReturnDetail
- CatReturnSummary
- CatTaskDetail
- DataPoint
- DataPointMetric
- DimensionsDetail
- IspDetail
- ProvinceDetail
- ResultSummary
- TaskAlarm

第 9 次发布

发布时间：2022-05-23 06:03:34

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- [CreateProbeTasks](#)
 - 新增入参：ClientNum
 - 新增出参：TaskIDs

修改数据结构：

- [NodeDefine](#)
 - 新增成员：CodeType, NodeDefineStatus

第 8 次发布

发布时间：2022-04-28 06:07:39

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- [CreateProbeTasks](#)
 - 新增入参：ProbeType, PluginSource

第 7 次发布

发布时间：2022-03-31 06:03:51

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [DescribeProbeMetricData](#)

第 6 次发布

发布时间：2022-01-18 08:04:56

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- [DescribeDetailedSingleProbeData](#)
 - 新增入参：City

第 5 次发布

发布时间：2021-12-22 08:07:25

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- [CreateProbeTasks](#)
 - 新增入参：Tag
- [DescribeProbeTasks](#)
 - 新增入参：TagFilters

新增数据结构：

- [KeyValuePair](#)
- [Tag](#)

修改数据结构：

- [ProbeTask](#)
 - 新增成员：TagInfoList

第 4 次发布

发布时间：2021-12-16 17:16:49

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [CreateProbeTasks](#)
- [DescribeProbeTasks](#)

新增数据结构：

- [ProbeTask](#)
- [ProbeTaskBasicConfiguration](#)

第 3 次发布

发布时间：2021-12-16 16:46:02

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [DeleteProbeTask](#)
- [DescribeDetailedSingleProbeData](#)
- [DescribeProbeNodes](#)
- [ResumeProbeTask](#)
- [SuspendProbeTask](#)
- [UpdateProbeTaskConfigurationList](#)

新增数据结构：

- [DetailedSingleDataDefine](#)
- [Field](#)
- [Label](#)
- [NodeDefine](#)
- [TaskResult](#)

第 2 次发布

发布时间：2020-01-10 14:24:52

本次发布包含了以下内容：

改善已有的文档。

删除接口：

- [CreateAlarmPloicy](#)
- [CreateTask](#)
- [DescribeAgentGroup](#)
- [DescribeAlarmGroups](#)
- [DescribeTask](#)
- [DescribeTasks](#)
- [GetRespTimeTrend](#)
- [ModifyAlarmPloicy](#)

- ModifyTask

删除数据结构:

- AlarmGroupInfo
- CatTask

第 1 次发布

发布时间: 2020-01-08 19:00:35

本次发布包含了以下内容:

改善已有的文档。

新增接口:

- BindAlarmPolicy
- CreateAgentGroup
- CreateAlarmPloicy
- CreateTask
- CreateTaskEx
- DeleteAgentGroup
- DeleteTasks
- DescribeAgentGroup
- DescribeAgentGroups
- DescribeAgents
- DescribeAlarmGroups
- DescribeAlarmTopic
- DescribeAlarms
- DescribeAlarmsByTask
- DescribeCatLogs
- DescribeTask
- DescribeTaskDetail
- DescribeTasks
- DescribeTasksByType
- DescribeUserLimit
- GetAvailRatioHistory
- GetDailyAvailRatio
- GetRealAvailRatio
- GetRespTimeTrend
- GetRespTimeTrendEx
- GetResultSummary
- GetReturnCodeHistory
- GetReturnCodeInfo
- GetTaskTotalNumber
- ModifyAgentGroup
- ModifyAlarmPloicy
- ModifyTask
- ModifyTaskEx
- PauseTask
- RunTask
- VerifyResult

新增数据结构:

- AgentGroup
- AlarmGroupInfo
- AlarmInfo
- AlarmTopic
- CatAgent

- CatLog
- CatReturnDetail
- CatReturnSummary
- CatTask
- CatTaskDetail
- DataPoint
- DataPointMetric
- DimensionsDetail
- IspDetail
- ProvinceDetail
- ResultSummary
- TaskAlarm

简介

最近更新时间：2021-06-23 11:29:52

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

欢迎使用云拨测（Cloud Automated Testing）。云拨测（Cloud Automated Testing，以下简称 CAT）利用分布于全球的服务质量监测点，对您的网站、域名、后台接口等进行周期性监控。用户可以使用本文档介绍的 API，并参照相应的示例，创建或修改拨测任务，查询拨测结果等。支持的全部操作可参见 API 概览。

API 概览

最近更新时间：2025-01-15 01:09:12

拨测资源（旧版）相关接口

接口名称	接口功能	频率限制（次/秒）
DescribeProbeNodes	查询拨测节点	20

云拨测（新版）相关接口

接口名称	接口功能	频率限制（次/秒）
CreateProbeTasks	批量创建拨测任务	20
DeleteProbeTask	删除任务	20
DescribeDetailedSingleProbeData	列出单次拨测详情数据	20
DescribeInstantTasks	获取历史即时拨测任务	20
DescribeNodes	获取拨测节点	20
DescribeProbeMetricData	列出云拨测指标详细数据	20
DescribeProbeTasks	分页查询拨测任务列表	20
ResumeProbeTask	恢复拨测任务	20
SuspendProbeTask	暂停任务	20
UpdateProbeTaskAttributes	更新探测任务属性	20
UpdateProbeTaskConfigurationList	批量更新拨测任务配置	20

注意：

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

调用方式

请求结构

最近更新時間：2024-08-28 01:11:50

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 `cat.tencentcloudapi.com`，也支持指定地域域名访问，例如广州地域的域名为 `cat.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 `cat.ap-guangzhou.tencentcloudapi.com` 是一致的。

注意：对时延敏感的业务，建议指定带地域的域名。

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

目前支持的域名列表为：

接入地域	域名
就近地域接入（推荐，只支持非金融区）	<code>cat.tencentcloudapi.com</code>
华南地区(广州)	<code>cat.ap-guangzhou.tencentcloudapi.com</code>
华东地区(上海)	<code>cat.ap-shanghai.tencentcloudapi.com</code>
华北地区(北京)	<code>cat.ap-beijing.tencentcloudapi.com</code>
西南地区(成都)	<code>cat.ap-chengdu.tencentcloudapi.com</code>
西南地区(重庆)	<code>cat.ap-chongqing.tencentcloudapi.com</code>
港澳台地区(中国香港)	<code>cat.ap-hongkong.tencentcloudapi.com</code>
亚太东南(新加坡)	<code>cat.ap-singapore.tencentcloudapi.com</code>
亚太东南(曼谷)	<code>cat.ap-bangkok.tencentcloudapi.com</code>
亚太南部(孟买)	<code>cat.ap-mumbai.tencentcloudapi.com</code>
亚太东北(首尔)	<code>cat.ap-seoul.tencentcloudapi.com</code>
亚太东北(东京)	<code>cat.ap-tokyo.tencentcloudapi.com</code>
美国东部(弗吉尼亚)	<code>cat.na-ashburn.tencentcloudapi.com</code>
美国西部(硅谷)	<code>cat.na-siliconvalley.tencentcloudapi.com</code>
欧洲地区(法兰克福)	<code>cat.eu-frankfurt.tencentcloudapi.com</code>

2. 通信协议

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

3. 请求方法

支持的 HTTP 请求方法：

- POST（推荐）
- GET

POST 请求支持的 Content-Type 类型：

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

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

4. 字符编码

均使用 UTF-8 编码。

公共参数

最近更新时间：2024-11-15 01:13:40

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输入公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
Authorization	String	是	HTTP 标准身份认证头部字段，例如： TC3-HMAC-SHA256 Credential=AKID***/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024 其中， - TC3-HMAC-SHA256：签名方法，目前固定取该值； - Credential：签名凭证，AKID*** 是 SecretId；Date 是 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致；service 为具体产品名，通常为域名前缀。例如，域名 cvm.tencentcloudapi.com 意味着产品名是 cvm。本产品取值为 cat；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。

参数名称	类型	必选	描述
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 01:14:11

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

推荐使用 API Explorer

</> 点击调试

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

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

为什么要进行签名

签名通过以下方式帮助保护请求：

1. 验证请求者的身份

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

2. 保护传输中的数据

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

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

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

申请安全凭证

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

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

申请安全凭证的具体步骤如下：

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

签名版本 v3 签名过程

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

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

1. 云服务器默认已开通，该接口很常用；
2. 该接口是只读的，不会改变现有资源的状态；
3. 接口覆盖的参数种类较全，可以演示包含数据结构的数组如何使用。

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

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

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

下面详细解释签名计算过程。

1. 拼接规范请求串

按如下伪代码格式拼接规范请求串 (CanonicalRequest)：

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

字段名称	解释
HTTPRequestMethod	HTTP 请求方法 (GET、POST)。此示例取值为 POST。
CanonicalURI	URI 参数, API 3.0 固定为正斜杠 (/)。
CanonicalQueryString	发起 HTTP 请求 URL 中的查询字符串, 对于 POST 请求, 固定为空字符串 "", 对于 GET 请求, 则为 URL 中间号 (?) 后面的字符串内容, 例如: Limit=10&Offset=0。 注意: CanonicalQueryString 需要参考 RFC3986 进行 URLEncode 编码 (特殊字符编码后需大写), 字符集 UTF-8。推荐使用编程语言标准库进行编码。
CanonicalHeaders	参与签名的头部信息, 至少包含 host 和 content-type 两个头部, 也可加入其他头部参与签名以提高自身请求的唯一性和安全性, 此示例额外增加了接口名头部。 拼接规则: 1. 头部 key 和 value 统一转成小写, 并去掉首尾空格, 按照 key:value\n 格式拼接; 2. 多个头部, 按照头部 key (小写) 的 ASCII 升序进行拼接。 此示例计算结果是 content-type:application/json; charset=utf-8\nhost:cvm.tencentcloudapi.com\nx-tc-action:describeinstances\n。 注意: content-type 必须和实际发送的相符合, 有些编程语言网络库即使未指定也会自动添加 charset 值, 如果签名时和发送时不一致, 服务器会返回签名校验失败。
SignedHeaders	参与签名的头部信息, 说明此次请求有哪些头部参与了签名, 和 CanonicalHeaders 包含的头部内容是一一对应的。content-type 和 host 为必选头部。 拼接规则: 1. 头部 key 统一转成小写; 2. 多个头部 key (小写) 按照 ASCII 升序进行拼接, 并且以分号 (;) 分隔。 此示例为 content-type;host;x-tc-action
HashedRequestPayload	请求正文 (payload, 即 body, 此示例为 {"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}) 的哈希值, 计算伪代码为 Lowercase(HexEncode(Hash.SHA256(RequestPayload))), 即对 HTTP 请求正文做 SHA256 哈希, 然后十六进制编

字段名称	解释
	码, 最后编码串转换成小写字母。对于 GET 请求, RequestPayload 固定为空字符串。此示例计算结果是 35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064。

根据以上规则, 示例中得到的规范请求串如下:

```
POST
/

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

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

2. 拼接待签名字符串

按如下格式拼接待签名字符串:

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

字段名称	解释
Algorithm	签名算法, 目前固定为 TC3-HMAC-SHA256。
RequestTimestamp	请求时间戳, 即请求头部的公共参数 X-TC-Timestamp 取值, 取当前时间 UNIX 时间戳, 精确到秒。此示例取值为 1551113065。
CredentialScope	凭证范围, 格式为 Date/service/tc3_request, 包含日期、所请求的服务和终止字符串 (tc3_request)。Date 为 UTC 标准时间的日期, 取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致; service 为产品名, 必须与调用的产品域名一致。此示例计算结果是 2019-02-25/cvm/tc3_request。
HashedCanonicalRequest	前述步骤拼接所得规范请求串的哈希值, 计算伪代码为 Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))。此示例计算结果是 7019a55be8395899b900fb5564e4200d984910f34794a27cb3fb7d10ff6a1e84。

注意:

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

根据以上规则, 示例中得到的待签名字符串如下:

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

3. 计算签名

1) 计算派生签名密钥，伪代码如下：

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

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

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

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

2) 计算签名，伪代码如下：

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

此示例计算结果是 10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f。

4. 拼接 Authorization

按如下格式拼接 Authorization：

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

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

根据以上规则，示例中得到的值为：

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

最终完整的调用信息如下：

```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKID*****/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host;x-tc-action, Signature=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f
Content-Type: application/json; charset=utf-8
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1551113065
X-TC-Region: ap-guangzhou

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

⚠ 注意:

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

签名演示

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

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

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

- [Signature Demo](#)

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

Java

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

public class TencentCloudAPITC3Demo {
    private final static Charset UTF8 = StandardCharsets.UTF_8;
    // 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****
    private final static String SECRET_ID = System.getenv("TENCENTCLOUD_SECRET_ID");
    // 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
}
```

```
private final static String SECRET_KEY = System.getenv("TENCENTCLOUD_SECRET_KEY");
private final static String CT_JSON = "application/json; charset=utf-8";

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

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

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

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

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

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

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

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

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

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

Python

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import hashlib, hmac, json, os, sys, time
from datetime import datetime

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

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

# ***** 步骤 1: 拼接规范请求串 *****
http_request_method = "POST"
canonical_uri = "/"
canonical_querystring = ""
```



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

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

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

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

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

Golang

```
package main

import (
"crypto/hmac"
```

```
"crypto/sha256"
"encoding/hex"
"fmt"
"os"
"strings"
"time"
)

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

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

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

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

```
timestamp,
credentialScope,
hashedCanonicalRequest)
fmt.Println(string2sign)

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

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

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

PHP

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

// step 1: build canonical request string
$httpRequestMethod = "POST";
$canonicalUri = "/";
$canonicalQueryString = "";
$canonicalHeaders = implode("\n", [
"content-type:application/json; charset=utf-8",
"host:".$host,
"x-tc-action:".strtolower($action),
```

```
""
]);
$signedHeaders = implode(";", [
"content-type",
"host",
"x-tc-action",
]);
$payload = '{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]';
$hashedRequestPayload = hash("SHA256", $payload);
$canonicalRequest = $httpRequestMethod."\n"
.$canonicalUri."\n"
.$canonicalQueryString."\n"
.$canonicalHeaders."\n"
.$signedHeaders."\n"
.$hashedRequestPayload;
echo $canonicalRequest.PHP_EOL;

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

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

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

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

Ruby

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

```
require 'json'
require 'time'
require 'openssl'

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

service = 'cvm'
host = 'cvm.tencentcloudapi.com'
endpoint = 'https://' + host
region = 'ap-guangzhou'
action = 'DescribeInstances'
version = '2017-03-12'
algorithm = 'TC3-HMAC-SHA256'
# timestamp = Time.now.to_i
timestamp = 1551113065
date = Time.at(timestamp).utc.strftime('%Y-%m-%d')

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

puts canonical_request

# ***** 步骤 2: 拼接待签名字符串 *****
credential_scope = date + '/' + service + '/' + 'tc3_request'
hashed_request_payload = Digest::SHA256.hexdigest(canonical_request)
string_to_sign = [
  algorithm,
  timestamp.to_s,
  credential_scope,
  hashed_request_payload,
].join("\n")
puts string_to_sign

# ***** 步骤 3: 计算签名 *****
digest = OpenSSL::Digest.new('sha256')
secret_date = OpenSSL::HMAC.digest(digest, 'TC3' + secret_key, date)
```

```
secret_service = OpenSSL::HMAC.digest(digest, secret_date, service)
secret_signing = OpenSSL::HMAC.digest(digest, secret_service, 'tc3_request')
signature = OpenSSL::HMAC.hexdigest(digest, secret_signing, string_to_sign)
puts signature

# ***** 步骤 4: 拼接 Authorization *****
authorization = "#{algorithm} Credential=#{secret_id}/#{credential_scope}, SignedHeaders=#{signed_headers}, Signature=#{signature}"
puts authorization

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

DotNet

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

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

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

    public static Dictionary<String, String> BuildHeaders(string secretid,
string secretkey, string service, string endpoint, string region,
string action, string version, DateTime date, string requestPayload)
    {
        string datestr = date.ToString("yyyy-MM-dd");
```

```
DateTime startTime = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc);
long requestTimestamp = (long)Math.Round((date - startTime).TotalMilliseconds, MidpointRounding.AwayFromZero) / 1000;
// ***** 步骤 1: 拼接规范请求串 *****
string algorithm = "TC3-HMAC-SHA256";
string httpRequestMethod = "POST";
string canonicalUri = "/";
string canonicalQueryString = "";
string contentType = "application/json";
string canonicalHeaders = "content-type:" + contentType + "; charset=utf-8\n"
+ "host:" + endpoint + "\n"
+ "x-tc-action:" + action.ToLower() + "\n";
string signedHeaders = "content-type;host;x-tc-action";
string hashedRequestPayload = SHA256Hex(requestPayload);
string canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload;
Console.WriteLine(canonicalRequest);

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

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

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

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

public static void Main(string[] args)
```

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

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

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

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

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

NodeJS

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

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

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

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

function main(){
```



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

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

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

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

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

// ***** 步骤 2: 拼接待签名字符串 *****
const algorithm = "TC3-HMAC-SHA256"
const hashedCanonicalRequest = getHash(canonicalRequest);
const credentialScope = date + "/" + service + "/" + "tc3_request"
const stringToSign = algorithm + "\n" +
timestamp + "\n" +
credentialScope + "\n" +
hashedCanonicalRequest
console.log(stringToSign)

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

// ***** 步骤 4: 拼接 Authorization *****
const authorization = algorithm + " " +
"Credential=" + SECRET_ID + "/" + credentialScope + ", " +
"SignedHeaders=" + signedHeaders + ", " +
```

```
"Signature=" + signature
console.log(authorization)

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

C++

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

using namespace std;

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

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

string sha256Hex(const string &str)
{
    char buf[3];
    unsigned char hash[SHA256_DIGEST_LENGTH];
    SHA256_CTX sha256;
    SHA256_Init(&sha256);
```

```
SHA256_Update(&sha256, str.c_str(), str.size());
SHA256_Final(hash, &sha256);
std::string NewString = "";
for(int i = 0; i < SHA256_DIGEST_LENGTH; i++)
{
    snprintf(buf, sizeof(buf), "%02x", hash[i]);
    NewString = NewString + buf;
}
return NewString;
}

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

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

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

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

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

    return (ss.str());
}

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

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

```
}
return output;
}

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

string service = "cvm";
string host = "cvm.tencentcloudapi.com";
string region = "ap-guangzhou";
string action = "DescribeInstances";
string version = "2017-03-12";
int64_t timestamp = 1551113065;
string date = get_data(timestamp);

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

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

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

// ***** 步骤 4: 拼接 Authorization *****
```

```
string authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + credentialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
cout << authorization << endl;

string curlcmd = "curl -X POST https://" + host + "\n"
+ " -H \"Authorization: \" + authorization + "\"\n"
+ " -H \"Content-Type: application/json; charset=utf-8\"\n" + "\n"
+ " -H \"Host: \" + host + "\"\n"
+ " -H \"X-TC-Action: \" + action + "\"\n"
+ " -H \"X-TC-Timestamp: \" + RequestTimestamp + "\"\n"
+ " -H \"X-TC-Version: \" + version + "\"\n"
+ " -H \"X-TC-Region: \" + region + "\"\n"
+ " -d '" + payload + "'";
cout << curlcmd << endl;
return 0;
};
```

C

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

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

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

void hmac_sha256(const char* key, int key_len,
const char* input, int input_len,
unsigned char* output, unsigned int* output_len)
{
    HMAC_CTX *h;
```

```
#if OPENSSSL_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 OPENSSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX_cleanup(h);
#else
HMAC_CTX_free(h);
#endif

}

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

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

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

int main()
{
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****
const char* SECRET_ID = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
const char* SECRET_KEY = getenv("TENCENTCLOUD_SECRET_KEY");
const char* service = "cvm";
const char* host = "cvm.tencentcloudapi.com";
const char* region = "ap-guangzhou";
const char* action = "DescribeInstances";
```

```
const char* version = "2017-03-12";
int64_t timestamp = 1551113065;
char date[20] = {0};
get_utc_date(timestamp, date, sizeof(date));

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

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

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

// ***** 步骤 3: 计算签名 *****
char k_key[64] = {0};
sprintf(k_key, "%s%s", "TC3", SECRET_KEY);
unsigned char k_date[64] = {0};
unsigned int output_len = 0;
hmac_sha256(k_key, strlen(k_key), date, strlen(date), k_date, &output_len);
unsigned char k_service[64] = {0};
hmac_sha256(k_date, output_len, service, strlen(service), k_service, &output_len);
unsigned char k_signing[64] = {0};
hmac_sha256(k_service, output_len, "tc3_request", strlen("tc3_request"), k_signing, &output_len);
unsigned char k_hmac_sha_sign[64] = {0};
hmac_sha256(k_signing, output_len, string_to_sign, strlen(string_to_sign), k_hmac_sha_sign, &output_len);

char signature[128] = {0};
```

```
hex_encode(k_hmac_sha_sign, output_len, signature);
printf("%s\n", signature);

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

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

其他语言

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

签名失败

存在以下签名失败的错误码，请根据实际情况处理。

错误码	错误描述
AuthFailure.SignatureExpire	签名过期。Timestamp 与服务器接收到请求的时间相差不得超过五分钟。
AuthFailure.SecretIdNotFound	密钥不存在。请到控制台查看密钥是否被禁用，是否少复制了字符或者多了字符。
AuthFailure.SignatureFailure	签名错误。可能是签名计算错误，或者签名与实际发送的内容不相符合，也有可能是密钥 SecretKey 错误导致的。
AuthFailure.TokenFailure	临时证书 Token 错误。
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）。

签名方法

最近更新时间：2024-12-25 01:14:12

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

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

推荐使用 API Explorer

</> 点击调试

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

腾讯云 API 会对每个访问请求进行身份验证，即每个请求都需要在公共请求参数中包含签名信息（Signature）以验证请求者身份。

签名信息由安全凭证生成，安全凭证包括 SecretId 和 SecretKey；若用户还没有安全凭证，请前往 [云API密钥页面](#) 申请，否则无法调用云 API 接口。

1. 申请安全凭证

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

安全凭证包括 SecretId 和 SecretKey：

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

申请安全凭证的具体步骤如下：

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

注意：每个账号最多可以拥有两对 SecretId/SecretKey。

2. 生成签名串

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

假设用户的 SecretId 和 SecretKey 分别是：

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

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

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

参数名称	中文	参数值
Action	方法名	DescribeInstances
SecretId	密钥 ID	AKID*****
Timestamp	当前时间戳	1465185768
Nonce	随机正整数	11886
Region	实例所在区域	ap-guangzhou
InstanceIds.0	待查询的实例 ID	ins-09dx96dg
Offset	偏移量	0
Limit	最大允许输出	20
Version	接口版本号	2017-03-12

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

2.1. 对参数排序

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

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

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

2.2. 拼接请求字符串

此步骤生成请求字符串。

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

注意：“参数值”为原始值而非 url 编码后的值。

然后将格式化后的各个参数用"&"拼接在一起，最终生成的请求字符串为：

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

2.3. 拼接签名原文字符串

此步骤生成签名原文字符串。

签名原文字符串由以下几个参数构成：

1. 请求方法: 支持 POST 和 GET 方式，这里使用 GET 请求，注意方法为全大写。
2. 请求主机: 查看实例列表(DescribeInstances)的请求域名为: cvm.tencentcloudapi.com。实际的请求域名根据接口所属模块的不同而不同，详见各接口说明。
3. 请求路径: 当前版本云API的请求路径固定为 /。
4. 请求字符串: 即上一步生成的请求字符串。

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

示例的拼接结果为：

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

2.4. 生成签名串

此步骤生成签名串。

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

具体代码如下，以 PHP 语言为例：

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

```
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));  
echo $signStr;
```

最终得到的签名串为：

```
7RAM2xfNMO9EiVTNmPg06MRnCvQ=
```

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

3. 签名串编码

生成的签名串并不能直接作为请求参数，需要对其进行 URL 编码。

如上一步生成的签名串为 7RAM2xfNMO9EiVTNmPg06MRnCvQ=，最终得到的签名串请求参数（Signature）为：

7RAM2xfNMO9EiVTNmPg06MRnCvQ%3D，它将用于生成最终的请求 URL。

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

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

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

4. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理。

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

5. 签名演示

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

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

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

- [Signature Demo](#)

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

最终输出的 url 可能为：<https://cvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap->

```
guangzhou&SecretId=AKID*****&Signature=7RAM2xfNMO9EiVTNmPg06MRnCvQ%3D&Timestamp=1465185768&Version=2017-03-12。
```

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

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

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

Java

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

public class TencentCloudAPIDemo {
    private final static String CHARSET = "UTF-8";

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

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

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

    public static void main(String[] args) throws Exception {
        TreeMap<String, Object> params = new TreeMap<String, Object>(); // TreeMap可以自动排序
        // 实际调用时应当使用随机数，例如：params.put("Nonce", new Random().nextInt(java.lang.Integer.MAX_VALUE));
        params.put("Nonce", 11886); // 公共参数
        // 实际调用时应当使用系统当前时间，例如：params.put("Timestamp", System.currentTimeMillis() / 1000);
        params.put("Timestamp", 1465185768); // 公共参数
        // 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID*****
        params.put("SecretId", System.getenv("TENCENTCLOUD_SECRET_ID")); // 公共参数
    }
}
```

```
params.put("Action", "DescribeInstances"); // 公共参数
params.put("Version", "2017-03-12"); // 公共参数
params.put("Region", "ap-guangzhou"); // 公共参数
params.put("Limit", 20); // 业务参数
params.put("Offset", 0); // 业务参数
params.put("InstanceIds.0", "ins-09dx96dg"); // 业务参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
params.put("Signature", sign(getStringToSign(params), System.getenv("TENCENTCLOUD_SECRET_KEY"), "HmacSHA1")); // 公共参数
System.out.println(getUrl(params));
}
}
```

Python

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

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

import requests

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

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

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

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

```
# resp = requests.get("https://" + endpoint, params=data)
# print(resp.url)
```

Golang

```
package main

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

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

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

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

    for i := range keys {
        k := keys[i]
        buf.WriteString(k)
        buf.WriteString("=")
        buf.WriteString(params[k])
        buf.WriteString("&")
    }
    buf.Truncate(buf.Len() - 1)
```

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

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

PHP

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

ksort($params);

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

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

Ruby

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

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

```
secret_key = ENV["TENCENTCLOUD_SECRET_KEY"]

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

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

DotNet

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

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

  public static string MakeSignPlainText(SortedDictionary<string, string> requestParams, string requestMethod, string requestHost, string requestPath)
  {
    string retStr = "";
    retStr += requestMethod;
```



```
retStr += requestHost;
retStr += requestPath;
retStr += "?";
string v = "";
foreach (string key in requestParams.Keys)
{
    v += string.Format("{0}={1}&", key, requestParams[key]);
}
retStr += v.TrimEnd('&');
return retStr;
}

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

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

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

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

NodeJS

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

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

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

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

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

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

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

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

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

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

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

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

返回结果

最近更新時間：2024-03-12 01:12:31

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 06:17:17

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。

拨测资源（旧版）相关接口

查询拨测节点

最近更新時間：2025-03-20 10:53:29

1. 接口描述

接口請求域名：cat.tencentcloudapi.com。

查詢撥測節點

默認接口請求頻率限制：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

以下請求參數列表僅列出了接口請求參數和部分公共參數，完整公共參數列表見 [公共請求參數](#)。

參數名稱	必選	類型	描述
Action	是	String	公共參數 ，本接口取值：DescribeProbeNodes。
Version	是	String	公共參數 ，本接口取值：2018-04-09。
Region	否	String	公共參數 ，此參數為可選參數。
NodeType	否	Integer	節點類型 • 1 = IDC • 2 = LastMile • 3 = Mobile 示例值：1
Location	否	Integer	節點區域 • 1 = 中國大陸 • 2 = 港澳台 • 3 = 海外 示例值：1
IsIPv6	否	Boolean	是否IPv6 示例值：false
NodeName	否	String	名字模糊搜索 示例值：沈阳
PayMode	否	Integer	付費模式 • 1 = 試用版本 • 2 = 付費版本 示例值：1

3. 输出参数

參數名稱	類型	描述
NodeSet	Array of NodeDefine	節點列表 注意：此字段可能返回 null，表示取不到有效值。 示例值：[{"Code":1000,"Type":1,"Name":"北京"}]
RequestId	String	唯一請求 ID，由服務端生成，每次請求都會返回（若請求因其他原因未能抵達服務端，則該次請求不會獲得 RequestId）。定位問題時需要提供該次請求的 RequestId。

4. 示例

示例1 查询拨测节点

查询拨测节点

输入示例

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

{
  "NodeType": 0,
  "Location": 0,
  "IsIPv6": true,
  "NodeName": "广东",
  "PayMode": 0
}
```

输出示例

```
{
  "Response": {
    "NodeSet": [
      {
        "Name": "北京-北京市-中国电信[IDC]",
        "Code": "10000",
        "Type": 1,
        "NetService": "中国电信",
        "District": "北京",
        "City": "北京市",
        "IPType": 1,
        "Location": 1,
        "CodeType": "",
        "NodeDefineStatus": 1
      }
    ],
    "RequestId": "avb"
  }
}
```

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.DbQueryFailed	数据库查询错误。
InvalidParameter	参数错误。

云拨测（新版）相关接口

批量更新拨测任务配置

最近更新時間：2025-03-20 10:53:29

1. 接口描述

接口请求域名：cat.tencentcloudapi.com。

批量更新拨测任务配置

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：UpdateProbeTaskConfigurationList。
Version	是	String	公共参数，本接口取值：2018-04-09。
Region	否	String	公共参数，此参数为可选参数。
TaskIds.N	是	Array of String	任务 ID，如task-n1wchki8 示例值：task-xx
Nodes.N	是	Array of String	拨测节点，如10001，详细地区运营商拨测编号请联系云拨测。 示例值：["10001"]
Interval	是	Integer	拨测间隔，如30，单位为分钟。 示例值：30
Parameters	是	String	拨测参数，详细参数配置可参考云拨测官网文档。 示例值： { "ipType":0, "grabBag":0, "netcmpOn":1, "netcmpActiveX":0, "netcmpTimeout":20
Cron	否	String	定时任务cron表达式 示例值：* 0-5 * * *
ResourceIDs.N	否	Array of String	预付费套餐id 需要与taskId对应 示例值：["presub-xx"]
NodeIpType	否	Integer	拨测节点的IP类型，0-不限，1-IPv4，2-IPv6 示例值：0
BatchTasks.N	否	Array of ProbeTaskBasicConfiguration	批量任务名-地址 示例值：TargetAddress: "www.qq.com", Name: "test"

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 更新拨测任务配置

输入示例

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

{
  "TaskIds": [
    "task-xx"
  ],
  "Interval": 30,
  "Nodes": [
    "10001"
  ],
  "Parameters": "{\"ipType\":0,\"netIcmpOn\":1,\"netIcmpActive\":0,\"netIcmpTimeout\":20,\"netIcmpInterval\":\"0.5\",\"netIcmpNum\":4,\"netIcmpSize\":4,\"netIcmpDataCut\":1,\"netDnsOn\":0,\"netTracertOn\":1,\"netTracertTimeout\":20,\"netTracertNum\":0}",
  "Cron": "* 0-6 * * *"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "720b3231-5a85-4f05-aaab-c9b9596xxxxx"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
FailedOperation	操作失败。
FailedOperation.DbQueryFailed	数据库查询错误。
FailedOperation.DbRecordCreateFailed	数据库创建失败。
FailedOperation.DbRecordUpdateFailed	数据库更新失败。
FailedOperation.NoValidNodes	无有效节点。
FailedOperation.PreResourceIDFailed	预付费资源id绑定错误。
FailedOperation.ResourceNotFound	资源不存在。
FailedOperation.TaskNotRunning	任务未运行。
FailedOperation.TaskNotSuspended	任务未暂停。
FailedOperation.TaskOperationNotAllow	任务状态不允许当前操作。
FailedOperation.TaskTypeNotSame	批量拨测任务的类型不相同。
InternalServerError	内部错误。
InvalidParameter	参数错误。

暂停任务

最近更新时间：2025-03-20 10:53:30

1. 接口描述

接口请求域名：cat.tencentcloudapi.com。

暂停任务

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：SuspendProbeTask。
Version	是	String	公共参数 ，本接口取值：2018-04-09。
Region	否	String	公共参数 ，此参数为可选参数。
TaskIds.N	是	Array of String	任务 ID 示例值：["task-xx"]

3. 输出参数

参数名称	类型	描述
Total	Integer	任务总量 示例值：1
SuccessCount	Integer	任务成功量 注意：此字段可能返回 null，表示取不到有效值。 示例值：1
Results	Array of TaskResult	任务执行结果 注意：此字段可能返回 null，表示取不到有效值。
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 暂停任务

输入示例

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

{
  "TaskIds": [
    "task-xxx"
  ]
}
```

```
]
}
```

输出示例

```
{
  "Response": {
    "SuccessCount": 1,
    "Total": 1,
    "Results": [
      {
        "ErrorMessage": "",
        "Success": true,
        "TaskId": "task-xxx"
      }
    ],
    "RequestId": "720b3231-5a85-4f05-aaab-c9b9596xxxxx"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
FailedOperation	操作失败。
FailedOperation.DbQueryFailed	数据库查询错误。
FailedOperation.DbRecordCreateFailed	数据库创建失败。
FailedOperation.DbRecordUpdateFailed	数据库更新失败。
FailedOperation.ResourceNotFound	资源不存在。

错误码	描述
FailedOperation.TaskNotRunning	任务未运行。
FailedOperation.TaskNotSuspended	任务未暂停。
FailedOperation.TaskOperationNotAllow	任务状态不允许当前操作。
InternalError	内部错误。
InvalidParameter	参数错误。

恢复拨测任务

最近更新时间：2025-03-20 10:53:30

1. 接口描述

接口请求域名：cat.tencentcloudapi.com。

恢复拨测任务

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：ResumeProbeTask。
Version	是	String	公共参数 ，本接口取值：2018-04-09。
Region	否	String	公共参数 ，此参数为可选参数。
TaskIds.N	是	Array of String	任务 ID 示例值：["task-xx"]

3. 输出参数

参数名称	类型	描述
Total	Integer	任务总量 示例值：1
SuccessCount	Integer	任务成功量 注意：此字段可能返回 null，表示取不到有效值。 示例值：1
Results	Array of TaskResult	任务执行详情 注意：此字段可能返回 null，表示取不到有效值。
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 恢复拨测任务

输入示例

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

{
  "TaskIds": [
    "task-xx"
  ]
}
```

```
]
}
```

输出示例

```
{
  "Response": {
    "SuccessCount": 0,
    "Total": 0,
    "Results": [
      {
        "ErrorMessage": "",
        "Success": true,
        "TaskId": "task-xx"
      }
    ],
    "RequestId": "720b3231-5a85-4f05-aaab-c9b9596xxxxx"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
FailedOperation	操作失败。
FailedOperation.DbQueryFailed	数据库查询错误。
FailedOperation.DbRecordCreateFailed	数据库创建失败。
FailedOperation.DbRecordUpdateFailed	数据库更新失败。
FailedOperation.ErrPrePaidResourceExpire	任务绑定的预付费套餐已过期。

错误码	描述
FailedOperation.NoValidNodes	无有效节点。
FailedOperation.ResourceNotFound	资源不存在。
FailedOperation.TaskNotRunning	任务未运行。
FailedOperation.TaskNotSuspended	任务未暂停。
FailedOperation.TaskOperationNotAllow	任务状态不允许当前操作。
InternalError	内部错误。
InvalidParameter	参数错误。

列出单次拨测详情数据

最近更新时间：2025-03-20 10:53:31

1. 接口描述

接口请求域名：cat.tencentcloudapi.com。

根据时间范围、任务ID、运营商等条件查询单次拨测详情数据

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

</> 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数 ，本接口取值：DescribeDetailedSingleProbeData。
Version	是	String	公共参数 ，本接口取值：2018-04-09。
Region	否	String	公共参数 ，此参数为可选参数。
BeginTime	是	Integer	开始时间戳（毫秒级） 示例值：1000000000
EndTime	是	Integer	结束时间戳（毫秒级） 示例值：1000000000
TaskType	是	String	任务类型 AnalyzeTaskType_Network：网络质量 AnalyzeTaskType_Browse：页面性能 AnalyzeTaskType_UploadDownload：文件传输（含文件上传、文件下载） AnalyzeTaskType_Transport：端口性能 AnalyzeTaskType_MediaStream：音视频体验 示例值：AnalyzeTaskType_UploadDownload
SortField	是	String	待排序字段 可以填写 ProbeTime 拨测时间排序 也可填写 SelectedFields 中的选中字段 示例值：ProbeTime
Ascending	是	Boolean	true表示升序 示例值：true
SelectedFields.N	是	Array of String	选中字段，如ProbeTime、TransferTime、TransferSize等。 示例值：[ProbeTime]
Offset	是	Integer	起始取数位置 示例值：0
Limit	是	Integer	取数数量 示例值：10
TaskID.N	否	Array of String	任务ID 示例值：[task_fasdfs]
Operators.N	否	Array of String	拨测点运营商 这里实际按拨测结果中的运营商来填写即可 电信：中国电信 移动：中国移动

参数名称	必选	类型	描述
			联通：中国联通 示例值：[电信]
Districts.N	否	Array of String	拨测点地区 这里实际按拨测结果中的地区来填写即可 国内一般是省级单位，如广东、广西、中国香港；直辖市则填北京、上海 境外一般是国家名，如澳大利亚、新加坡 示例值：[上海]
ErrorTypes.N	否	Array of String	错误类型 示例值：[正常]
City.N	否	Array of String	城市 这里实际按拨测结果中的城市来填写即可 示例： 深圳市 武汉市 首尔 多伦多 示例值：[深圳市]
ScrollID	否	String	es scroll查询id 示例值： DXF1ZXJ5QW5kRmV0Y2gBAAAAQEPczgWVU1RMC12VHhSUWIya1I0dnU4aXlvUQ==
QueryFlag	否	String	详情数据下载 示例值："download"

3. 输出参数

参数名称	类型	描述
DataSet	Array of DetailedSingleDataDefine	单次详情数据
TotalNumber	Integer	符合条件的数据总数 示例值：10000
ScrollID	String	es scroll查询的id 示例值： DXF1ZXJ5QW5kRmV0Y2gBAAAAQEPczgWVU1RMC12VHhSUWIya1I0dnU4aXlvUQ==
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 列出单次拨测详情数据

列出单次拨测详情数据

输入示例

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

```
{
```

```
"BeginTime": 167989994107,
"EndTime": 1679903594107,
"TaskType": "AnalyzeTaskType_Network",
"SortField": "ProbeTime",
"Ascending": true,
"Offset": 0,
"Limit": 10,
"SelectedFields": [
  "ProbeTime",
  "City",
  "Operator",
  "ProbeIP"
],
"TaskID": [
  "task-p4najqd4"
],
"ScrollID": "DXF1ZXJ5QW5kRmV0Y2gBAAAAAQEPczgWVU1RMC12VHhSUW1Ya1I0dnU4aXlxxx=="
}
```

输出示例

```
{
  "Response": {
    "ScrollID": "DXF1ZXJ5QW5kRmV0Y2gBAAAAAQEPczgWVU1RMC12VHhSUW1Ya1I0dnU4aXlxxx==",
    "TotalNumber": 10,
    "DataSet": [
      {
        "ProbeTime": 1,
        "Labels": [
          {
            "ID": 0,
            "Name": "City",
            "Value": "上海市"
          }
        ],
        "Fields": [
          {
            "ID": 0,
            "Name": "ProbeTime",
            "Value": 1679903537000
          }
        ]
      }
    ],
    "RequestId": "720b3231-5a85-4f05-aaab-c9b9596xxxxx"
  }
}
```

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.DbQueryFailed	数据库查询错误。
FailedOperation.ESQueryError	ES查询错误。
InternalServerError	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。
UnknownParameter	未知参数错误。

删除任务

最近更新时间：2025-03-20 10:53:31

1. 接口描述

接口请求域名：cat.tencentcloudapi.com。

删除拨测任务

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：DeleteProbeTask。
Version	是	String	公共参数 ，本接口取值：2018-04-09。
Region	否	String	公共参数 ，此参数为可选参数。
TaskIds.N	是	Array of String	任务 ID 示例值：["task-xx"]

3. 输出参数

参数名称	类型	描述
Total	Integer	任务总量 示例值：1
SuccessCount	Integer	任务成功量 注意：此字段可能返回 null，表示取不到有效值。 示例值：1
Results	Array of TaskResult	任务执行结果 注意：此字段可能返回 null，表示取不到有效值。
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 删除任务

输入示例

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

{
  "TaskIds": [
    "task-xxx"
  ]
}
```

```
]
}
```

输出示例

```
{
  "Response": {
    "SuccessCount": 0,
    "Total": 0,
    "Results": [
      {
        "ErrorMessage": "",
        "Success": true,
        "TaskId": "task-xxx"
      }
    ],
    "RequestId": "720b3231-5a85-4f05-aaab-c9b9596xxxxx"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
FailedOperation	操作失败。
FailedOperation.DbQueryFailed	数据库查询错误。
FailedOperation.DbRecordCreateFailed	数据库创建失败。
FailedOperation.DbRecordUpdateFailed	数据库更新失败。
FailedOperation.NoValidNodes	无有效节点。

错误码	描述
FailedOperation.ResourceNotFound	资源不存在。
FailedOperation.TaskNotRunning	任务未运行。
FailedOperation.TaskNotSuspended	任务未暂停。
FailedOperation.TaskOperationNotAllow	任务状态不允许当前操作。
InternalError	内部错误。
InvalidParameter	参数错误。

分页查询拨测任务列表

最近更新時間：2025-03-20 10:53:30

1. 接口描述

接口請求域名：cat.tencentcloudapi.com。

查詢撥測任務列表

默認接口請求頻率限制：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

以下請求參數列表僅列出了接口請求參數和部分公共參數，完整公共參數列表見 [公共請求參數](#)。

參數名稱	必選	類型	描述
Action	是	String	公共參數，本接口取值：DescribeProbeTasks。
Version	是	String	公共參數，本接口取值：2018-04-09。
Region	否	String	公共參數，此參數為可選參數。
TaskIDs.N	否	Array of String	任務 ID 列表 示例值：["task-xx"]
TaskName	否	String	任務名 示例值：probe
TargetAddress	否	String	撥測目標 示例值：http://www.qq.com
TaskStatus.N	否	Array of Integer	任務狀態列表 • 1 = 創建中 • 2 = 運行中 • 3 = 運行異常 • 4 = 暫停中 • 5 = 暫停異常 • 6 = 任務暫停 • 7 = 任務刪除中 • 8 = 任務刪除異常 • 9 = 任務刪除 • 10 = 定時任務暫停中 示例值：[1,2]
Offset	否	Integer	偏移量，默認為0 示例值：1
Limit	否	Integer	返回數量，默認為20，最大值為100 示例值：10
PayMode	否	Integer	付費模式 • 1 = 試用版本 • 2 = 付費版本 示例值：1
OrderState	否	Integer	訂單狀態 • 1 = 正常 • 2 = 欠費

参数名称	必选	类型	描述
			示例值: 1
TaskType.N	否	Array of Integer	拨测类型 <ul style="list-style-type: none"> • 1 = 页面浏览 • 2 = 文件上传 • 3 = 文件下载 • 4 = 端口性能 • 5 = 网络质量 • 6 = 流媒体 即使拨测只支持页面浏览, 网络质量, 文件下载 示例值: [1,2,3]
TaskCategory.N	否	Array of Integer	节点类型 示例值: [1,2,3]
OrderBy	否	String	排序的列 示例值: task_category
Ascend	否	Boolean	是否正序 示例值: true
TagFilters.N	否	Array of KeyValuePair	资源标签值 示例值: [{"Key": "name", "Value": "zhangsan"}]

3. 输出参数

参数名称	类型	描述
TaskSet	Array of ProbeTask	任务列表 注意: 此字段可能返回 null, 表示取不到有效值。
Total	Integer	任务总数 示例值: 3
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 示例2

输入示例

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

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

输出示例

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

```
"RequestId": "81a92494-0fb9-4b10-8ce4-bfa4565cf4eb",
"TaskSet": null,
"Total": 0
}
}
```

示例2 示例3

输入示例

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

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

输出示例

```
{
  "Response": {
    "RequestId": "11483de9-9be5-4537-a846-339a57f0e90e",
    "TaskSet": null,
    "Total": 0
  }
}
```

示例3 示例5

输入示例

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

{}
```

输出示例

```
{
  "Response": {
    "RequestId": "d7628289-fd3c-4acf-a140-a33dd02b3ff6",
    "TaskSet": null,
    "Total": 0
  }
}
```

示例4 查询拨测任务列表

输入示例

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

```
{
  "TaskIDs": [
    "task-xx"
  ],
  "TagFilters": [
    {
      "Key": "name",
      "Value": "zhangsan"
    }
  ]
}
```

输出示例

```
{
  "Response": {
    "TaskSet": [
      {
        "Name": "cron测试",
        "TaskId": "task-xx",
        "TaskType": 0,
        "Nodes": [
          "10000"
        ],
        "NodeIpType": 0,
        "Interval": 0,
        "Parameters": "{\"ipType\":0,\"netIcmpOn\":1,\"netIcmpActivex\":0,\"netIcmpTimeout\":20,\"netIcmpInterval\":\"0.5\",\"netIcmpNum\":4,\"netIcmpSize\":4,\"netIcmpDataCut\":1,\"netDnsOn\":0,\"netTracertOn\":1,\"netTracertTimeout\":20,\"netTracertNum\":0}",
        "Status": 0,
        "TargetAddress": "www.test.com",
        "PayMode": 0,
        "OrderState": 0,
        "TaskCategory": 0,
        "CreatedAt": "2024-01-02 16:57:15",
        "Cron": "* 0-6 * * *",
        "CronState": 0,
        "TagInfoList": [
          {
            "Key": "部门",
            "Value": "部门 1"
          }
        ],
        "Total": 0,
        "RequestId": "720b3231-5a85-4f05-aaab-c9b9596xxxxx"
      }
    ]
  }
}
```

```
}  
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
FailedOperation	操作失败。
FailedOperation.DbQueryFailed	数据库查询错误。
FailedOperation.DbRecordCreateFailed	数据库创建失败。
FailedOperation.DbRecordUpdateFailed	数据库更新失败。
FailedOperation.NoValidNodes	无有效节点。
FailedOperation.ResourceNotFound	资源不存在。
InternalError	内部错误。
InvalidParameter	参数错误。

批量创建拨测任务

最近更新時間：2025-03-20 10:53:32

1. 接口描述

接口请求域名：cat.tencentcloudapi.com。

批量创建拨测任务

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：CreateProbeTasks。
Version	是	String	公共参数 ，本接口取值：2018-04-09。
Region	否	String	公共参数 ，此参数为可选参数。
BatchTasks.N	是	Array of ProbeTaskBasicConfiguration	批量任务名-地址
TaskType	是	Integer	任务类型，如1、2、3、4、5、6、7；1-页面性能、2-文件上传、3-文件下载、4-端口性能、5-示例值：1
Nodes.N	是	Array of String	拨测节点，如10001，具体拨测地域运营商对应的拨测点编号可联系云拨测确认。 示例值：["10001"]
Interval	是	Integer	拨测间隔，单位为分钟 示例值：30
Parameters	是	String	拨测参数，详细可参考云拨测官方文档，链接： https://cloud.tencent.com/document/product/377/34711 示例值： { "ipType": 0, "grabBag": 0, "netclmpOn": 1, "netclmpActivex": 0, "netclmpTimeout": 20,
TaskCategory	是	Integer	任务分类 • 1 = PC • 2 = Mobile 示例值：1
Cron	否	String	定时任务cron表达式 示例值：* 0-5 * * *
Tag.N	否	Array of Tag	资源标签值 示例值：[{"TagKey": "运营部门", "TagValue": "云产一"}]
ProbeType	否	Integer	测试类型，包含定时测试与即时测试。0-定时拨测，其它表示即时拨测。 示例值：1
PluginSource	否	String	插件类型，如CDN，详情参考云拨测官方文档。 示例值：CDN
ClientNum	否	String	客户端ID 示例值：3198058
NodelpType	否	Integer	拨测点IP类型：0-不限制IP类型，1-IPv4，2-IPv6 示例值：0

参数名称	必选	类型	描述
SubSyncFlag	否	Integer	供应商子账户同步标志 示例值: 0

3. 输出参数

参数名称	类型	描述
TaskIDs	Array of String	任务ID列表 示例值: ["task-fsz5tuvy"]
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 示例1

输入示例

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

{
  "BatchTasks": [
    {
      "TargetAddress": "http://www.baidu.com",
      "Name": "CDN验证21"
    }
  ],
  "Parameters": "{\"ipType\":0,\"grabBag\":0,\"netIcmpOn\":1,\"netIcmpActivex\":0,\"netIcmpTimeout\":20,\"netIcmpInterval\":0.5,\"netIcmpNum\":20,\"netIcmpSize\":32,\"netIcmpDataCut\":1,\"netDnsOn\":1,\"netDnsTimeout\":5,\"netDnsQuerymethod\":1,\"netDnsNs\":\"\",\"netDigOn\":1,\"netDnsServer\":2,\"netTracertOn\":1,\"netTracertTimeout\":60,\"netTracertNum\":30,\"whiteList\":\"\",\"blackList\":\"\",\"netIcmpActivexStr\":\"\"}",
  "TaskCategory": "1",
  "Interval": "5",
  "TaskType": "5",
  "Nodes": [
    "12136"
  ]
}
```

输出示例

```
{
  "Response": {
    "RequestId": "0c4977a0-5b9f-469b-84d4-5d6a609e93a3",
    "TaskIDs": [
      "task-h4bcnook"
    ]
  }
}
```

示例2 批量创建拨测任务

输入示例

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

{
  "BatchTasks": [
    {
      "Name": "probe",
      "TargetAddress": "http://www.baidu.com"
    }
  ],
  "Parameters": "{\n\"ipType\":0,\n\"grabBag\":0,\n\"netIcmpOn\":1,\n\"netIcmpActivex\":0,\n\"netIcmpTimeout\":20,\n\"netIcmpInterval\":0.5,\n\"netIcmpNum\":20,\n\"netIcmpSize\":32,\n\"netIcmpDataCut\":1,\n\"netDnsOn\":1,\n\"netDnsTimeout\":5,\n\"netDnsQuerymethod\":1,\n\"netDnsNs\":\"\n\", \n\"netDigOn\":1,\n\"netDnsServer\":2,\n\"netTracertOn\":1,\n\"netTracertTimeout\":60,\n\"netTracertNum\":30,\n\"whiteList\":\"\n\", \n\"blackList\":\"\n\", \n\"netIcmpActivexStr\":\"\n\"}",
  "Interval": 30,
  "TaskCategory": 1,
  "TaskType": 5,
  "Nodes": [
    "10001"
  ],
  "Cron": "* * 0-6 * * *",
  "Tag": [
    {
      "TagKey": "运营部门",
      "TagValue": "产一"
    }
  ]
}
```

输出示例

```
{
  "Response": {
    "TaskIDs": [
      "task-abc"
    ],
    "RequestId": "720b3231-5a85-4f05-aaab-c9b9596xxxxxx"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
FailedOperation	操作失败。
FailedOperation.DbQueryFailed	数据库查询错误。
FailedOperation.DbRecordCreateFailed	数据库创建失败。
FailedOperation.DbRecordUpdateFailed	数据库更新失败。
FailedOperation.ErrPrePaidResourceExpire	任务绑定的预付费套餐已过期。
FailedOperation.NoValidNodes	无有效节点。
FailedOperation.OrderOutOfCredit	账单欠费。
FailedOperation.PreResourceIDFailed	预付费资源id绑定错误。
FailedOperation.ResourceNotFound	资源不存在。
FailedOperation.SendRequest	发送授权请求失败。
FailedOperation.TagRequiredVerifyFailed	标签验证错误。
FailedOperation.TaskNotRunning	任务未运行。
FailedOperation.TaskNotSuspended	任务未暂停。
FailedOperation.TaskOperationNotAllowed	任务状态不允许当前操作。
FailedOperation.TaskTypeNotSame	批量拨测任务的类型不相同。
FailedOperation.TrialTaskExceed	试用任务量超时。
FailedOperation.UnmarshalResponse	json解析失败。
FailedOperation.UserNoQcloudTAGFullAccess	鉴权失败。
InternalError	内部错误。
InvalidParameter	参数错误。

列出云拨测指标详细数据

最近更新时间：2025-03-20 10:53:30

1. 接口描述

接口请求域名：cat.tencentcloudapi.com。

查询云拨测指标数据，指标支持使用sum,avg,max,min聚合函数进行指标数据查询

拨测频率与groupby聚合时间设置关联，如拨测频率为 30 分钟，则 groupby 聚合时间建议设置为大于30分钟，避免出现查询数据为空的情况

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：DescribeProbeMetricData。
Version	是	String	公共参数 ，本接口取值：2018-04-09。
Region	否	String	公共参数 ，此参数为可选参数。
AnalyzeTaskType	否	String	分析任务类型，支持以下几种类型： AnalyzeTaskType_Network：网络质量 AnalyzeTaskType_Browse：页面性能 AnalyzeTaskType_Transport：端口性能 AnalyzeTaskType_UploadDownload：文件传输 AnalyzeTaskType_MediaStream：音视频体验 示例值：AnalyzeTaskType_UploadDownload
MetricType	否	String	指标类型（counter、gauge以及histogram），指标查询默认传gauge 示例值：gauge
Field	否	String	指标详细字段，可以传递具体的指标也可以对指标进行聚合查询例如："avg(ping_time)"代表整体时延(ms)；不同的任务类型支持不同的field查询，以及聚合规则，详情可见 https://cloud.tencent.com/document/product/248/87584 。 示例值：avg(ping_time)
Filter	否	String	过滤条件可以传单个过滤条件也可以拼接多个参数 示例值：host = '123' and time >= now()-30m
GroupBy	否	String	聚合时间, 1m、1d、30d 等等 示例值：time(1m), operator
Filters.N	否	Array of String	多条件过滤，支持多个过滤条件组合查询 例如：["host" = "www.test.com", "time >= now()-1h"] 示例值：["host" = "https://test.com", "time >= now()-1h"]

3. 输出参数

参数名称	类型	描述
MetricSet	String	返回指标 JSON 序列化后的字符串，具体如下所示：{"name":"task_navigate_request_gauge","columns":["time","avg(first_screen_time) / 1000"],"values":[[1641571200,6.756600000000001]],"tags":null} 示例值：gauge
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 请求浏览数据

请求浏览数据

输入示例

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

{
  "AnalyzeTaskType": "AnalyzeTaskType_Network",
  "MetricType": "gauge",
  "Field": "sum(\"error_count\")",
  "GroupBy": "time(30d), \"operator\"",
  "Filters": [
    "\"host\" = 'default-host'",
    "time >= now()-1h"
  ]
}
```

输出示例

```
{
  "Response": {
    "MetricSet": "[{\"name\": \"netenv_request_gauge\", \"columns\": [\"time\", \"sum(error_count)\"], \"values\": [[xxxxx, xxxxx], \"tags\": null}]",
    "RequestId": "720b3231-5a85-4f05-aaab-c9b9596xxxxx"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
FailedOperation	操作失败。
FailedOperation.DbQueryFailed	数据库查询错误。

获取拨测节点

最近更新时间：2025-03-20 10:53:31

1. 接口描述

接口请求域名：cat.tencentcloudapi.com。

获取拨测节点

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：DescribeNodes。
Version	是	String	公共参数 ，本接口取值：2018-04-09。
Region	否	String	公共参数 ，此参数为可选参数。
NodeType	否	Integer	节点类型 • 1 = IDC • 2 = LastMile • 3 = Mobile 示例值：1
Location	否	Integer	节点区域 • 1 = 中国大陆 • 2 = 港澳台 • 3 = 境外 示例值：1
IsIPv6	否	Boolean	是否IPv6 示例值：false
NodeName	否	String	名字模糊搜索 示例值：沈阳
PayMode	否	Integer	付费模式 • 1 = 试用版本 • 2 = 付费版本 示例值：1
TaskType	否	Integer	任务类型 • 1 = 页面性能 • 2 = 文件上传 • 3 = 文件下载 • 4 = 端口性能 • 5 = 网络质量 • 6 = 音视频体验 示例值：1

3. 输出参数

参数名称	类型	描述
NodeSet	Array of NodeDefineExt	节点列表 注意：此字段可能返回 null，表示取不到有效值。 示例值：[{"Code":1000, "Type":1, "Name":"北京"}]
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 获取拨测节点

获取拨测节点

输入示例

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

{
  "NodeType": 1
}
```

输出示例

```
{
  "Response": {
    "NodeSet": [
      {
        "City": "北京",
        "Code": "10001",
        "Name": "test_task",
        "District": "beijing",
        "NetService": "中国电信",
        "Location": 0,
        "Type": 1,
        "IPType": 0,
        "TaskTypes": [
          1,
          2
        ],
        "CodeType": ""
      }
    ],
    "RequestId": "720b3231-5a85-4f05-aaab-c9b9596xxxxx"
  }
}
```

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](#)。

更新探测任务属性

最近更新时间：2025-03-20 10:53:29

1. 接口描述

接口请求域名：cat.tencentcloudapi.com。

更新探测任务属性

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：UpdateProbeTaskAttributes。
Version	是	String	公共参数 ，本接口取值：2018-04-09。
Region	否	String	公共参数 ，此参数为可选参数。
TaskId	是	String	任务 ID 示例值：task-xx
Name	否	String	任务名，该参数为空时不作任何修改。 示例值：update_task_attributes

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 更新探测任务属性

输入示例

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

{
  "TaskId": "task-a3a9qbqy",
  "Name": "qq_Test"
}
```

输出示例


```
{
  "Response": {
    "RequestId": "db996865-85af-496f-b081-3e2fff7e5af2"
  }
}
```

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.DbQueryFailed	数据库查询错误。
FailedOperation.DbRecordCreateFailed	数据库创建失败。
FailedOperation.DbRecordUpdateFailed	数据库更新失败。
FailedOperation.NoValidNodes	无有效节点。
FailedOperation.ResourceNotFound	资源不存在。
FailedOperation.TaskNotRunning	任务未运行。
FailedOperation.TaskNotSuspended	任务未暂停。
FailedOperation.TaskOperationNotAllowed	任务状态不允许当前操作。
InternalServerError	内部错误。
InvalidParameter	参数错误。

获取历史即时拨测任务

最近更新時間：2025-03-20 10:53:31

1. 接口描述

接口請求域名：cat.tencentcloudapi.com。

獲取歷史即時撥測任務

默認接口請求頻率限制：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

以下請求參數列表僅列出了接口請求參數和部分公共參數，完整公共參數列表見 [公共請求參數](#)。

參數名稱	必選	類型	描述
Action	是	String	公共參數 ，本接口取值：DescribeInstantTasks。
Version	是	String	公共參數 ，本接口取值：2018-04-09。
Region	否	String	公共參數 ，此參數為可選參數。
Limit	是	Integer	數量 示例值：10
Offset	是	Integer	起始位置 示例值：1

3. 输出参数

參數名稱	類型	描述
Tasks	Array of SingleInstantTask	任務 注意：此字段可能返回 null，表示取不到有效值。
Total	Integer	總數 示例值：10
RequestId	String	唯一請求 ID，由服務端生成，每次請求都會返回（若請求因其他原因未能抵達服務端，則該次請求不會獲得 RequestId）。定位問題時需要提供該次請求的 RequestId。

4. 示例

示例1 获取历史即时拨测任务

輸入示例

```
POST / HTTP/1.1
Host: cat.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DescribeInstantTasks
<公共請求參數>

{
  "Limit": 1,
```

```
"Offset": 1
}
```

输出示例

```
{
  "Response": {
    "Tasks": [
      {
        "Status": "1",
        "ProbeTime": "1641917822000",
        "TaskCategory": 1,
        "SuccessRate": 0.0,
        "TargetAddress": "www.test.com",
        "TaskId": "task-xxx",
        "TaskType": 1,
        "NodeCount": 1
      }
    ],
    "RequestId": "720b3231-5a85-4f05-aaab-c9b9596xxxxx",
    "Total": 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

该接口暂无业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

数据结构

最近更新时间：2025-03-07 01:09:27

DetailedSingleDataDefine

单条详细拨测数据

被如下接口引用：DescribeDetailedSingleProbeData。

名称	类型	描述
ProbeTime	Integer	拨测时间戳 示例值：1000000000
Labels	Array of Label	储存所有string类型字段 示例值：[{...}, {...}]
Fields	Array of Field	储存所有float类型字段 示例值：[{...},{...}]

Field

储存float类型字段

被如下接口引用：DescribeDetailedSingleProbeData。

名称	类型	描述
ID	Integer	自定义字段编号 示例值：1
Name	String	自定义字段名称/说明 示例值：probeTime
Value	Float	字段值 示例值：10000000000

KeyValuePair

键值对

被如下接口引用：DescribeProbeTasks。

名称	类型	必选	描述
Key	String	是	键 示例值：Name
Value	String	是	值 示例值：zhangsan

Label

保存string类型字段

被如下接口引用：DescribeDetailedSingleProbeData。

名称	类型	描述
ID	Integer	自定义字段编号 示例值：1
Name	String	自定义字段名称/说明 示例值：probeIP
Value	String	字段值

名称	类型	描述
		示例值: 0.0.0.0

NodeDefine

探测节点

被如下接口引用: DescribeProbeNodes。

名称	类型	描述
Name	String	节点名称 示例值: 中国电信-北京 IDC
Code	String	节点代码 示例值: 10001
Type	Integer	节点类型 <ul style="list-style-type: none"> 1 = IDC 2 = LastMile 3 = Mobile 示例值: 1
NetService	String	网络服务商 示例值: 中国移动
District	String	区域 示例值: 福建
City	String	城市 示例值: 宁德市
IPType	Integer	IP 类型 <ul style="list-style-type: none"> 1 = IPv4 2 = IPv6 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: 1
Location	Integer	区域 <ul style="list-style-type: none"> 1 = 中国大陆 2 = 港澳台 3 = 国外 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: 1
CodeType	String	节点类型 如果为base 则为可用性拨测点, 为空则为高级拨测点 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: base
NodeDefineStatus	Integer	节点状态: 1-运行,2-下线 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: 1

NodeDefineExt

探测节点

被如下接口引用: DescribeNodes。

名称	类型	描述
Name	String	节点名称 示例值: 中国电信-北京 IDC
Code	String	节点代码

名称	类型	描述
		示例值: 10001
Type	Integer	节点类型 <ul style="list-style-type: none"> • 1 = IDC • 2 = LastMile • 3 = Mobile 示例值: 1
NetService	String	网络服务商 示例值: 中国移动
District	String	区域 示例值: 福建
City	String	城市 示例值: 宁德市
IPType	Integer	IP 类型 <ul style="list-style-type: none"> • 1 = IPv4 • 2 = IPv6 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: 1
Location	Integer	区域 <ul style="list-style-type: none"> • 1 = 中国大陆 • 2 = 港澳台 • 3 = 境外 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: 1
CodeType	String	节点类型 如果为base 则为可用性拨测点, 为空则为高级拨测点 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: base
TaskTypes	Array of Integer	节点支持的任务类型。1: 页面性能 2: 文件上传 3: 文件下载 4: 端口性能 5: 网络质量 6: 音视频体验 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: [1, 2, 3]

ProbeTask

拨测任务

被如下接口引用: DescribeProbeTasks。

名称	类型	描述
Name	String	任务名 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: probe
TaskId	String	任务 ID 示例值: task-xx
TaskType	Integer	拨测类型 <ul style="list-style-type: none"> • 1 = 页面浏览 • 2 = 文件上传 • 3 = 文件下载 • 4 = 端口性能 • 5 = 网络质量 • 6 = 流媒体 即时拨测只支持页面浏览, 网络质量, 文件下载 示例值: 1

名称	类型	描述
Nodes	Array of String	拨测节点列表 示例值: ["10000"]
NodelpType	Integer	拨测任务所选的拨测点IP类型, 0-不限, 1-IPv4, 2-IPv6 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: 0
Interval	Integer	拨测间隔, 单位为分钟 示例值: 30
Parameters	String	拨测参数 示例值: { "ipType":0, "grabBag":0, "netlcmpOn":1, "netlcmpActiveX":0, "netlcmpTimeout":20, "netlcmpInterval":0.5, "r
Status	Integer	任务状态 <ul style="list-style-type: none"> 1 = 创建中 2 = 运行中 3 = 运行异常 4 = 暂停中 5 = 暂停异常 6 = 任务暂停 7 = 任务删除中 8 = 任务删除异常 9 = 任务删除 10 = 定时任务暂停中 示例值: 1
TargetAddress	String	目标地址 示例值: http://www.qq.com
PayMode	Integer	付费模式 <ul style="list-style-type: none"> 1 = 试用版本 2 = 付费版本 示例值: 1
OrderState	Integer	订单状态 <ul style="list-style-type: none"> 1 = 正常 2 = 欠费 示例值: 1
TaskCategory	Integer	任务分类 <ul style="list-style-type: none"> 1 = PC 2 = Mobile 示例值: 1
CreatedAt	String	创建时间 示例值: "2020"
Cron	String	定时任务cron表达式 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: * 0-5 * * *
CronState	Integer	定时任务启动状态 <ul style="list-style-type: none"> 1 = 定时任务表达式生效 2 = 定时任务表达式未生效 (一般为任务手动暂停) 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: 1
TagInfoList	Array of KeyValuePair	任务当前绑定的标签 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: [{"Key": "name", "Value": "zhangsan"}]
SubSyncFlag	Integer	是否为同步账号 注意: 此字段可能返回 null, 表示取不到有效值。

名称	类型	描述
		示例值：同步标记

ProbeTaskBasicConfiguration

拨测任务基础配置

被如下接口引用：CreateProbeTasks, UpdateProbeTaskConfigurationList。

名称	类型	必选	描述
Name	String	是	拨测任务名称 示例值：probe-task1
TargetAddress	String	是	拨测目标地址 示例值：www.baidu.com

SingleInstantTask

单个即时拨测任务信息

被如下接口引用：DescribeInstantTasks。

名称	类型	描述
TaskId	String	任务ID 示例值：task-123
TargetAddress	String	任务地址 示例值：http://qq.com
TaskType	Integer	任务类型 示例值：5
ProbeTime	Integer	测试时间 示例值：1641917822000
Status	String	任务状态 示例值：成功
SuccessRate	Float	成功率 示例值：0.8
NodeCount	Integer	节点数量 示例值：1
TaskCategory	Integer	节点类型 示例值：1

Tag

资源的标签，通过标签对资源进行划分用于支持细粒度的鉴权、分账等场景

被如下接口引用：CreateProbeTasks。

名称	类型	必选	描述
TagKey	String	是	key 示例值：k
TagValue	String	是	value 示例值：v

TaskResult

任务执行结果

被如下接口引用: DeleteProbeTask, ResumeProbeTask, SuspendProbeTask。

名称	类型	描述
TaskId	String	任务 ID 示例值: task-xx
Success	Boolean	是否成功 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: true
ErrorMessage	String	错误信息 注意: 此字段可能返回 null, 表示取不到有效值。 示例值: 错误信息

错误码

最近更新時間：2024-12-20 01:09:41

功能說明

如果返回結果中存在 Error 字段，則表示調用 API 接口失敗。例如：

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

Error 中的 Code 表示錯誤碼，Message 表示該錯誤的具體信息。

錯誤碼列表

公共錯誤碼

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

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

业务错误码

错误码	说明
FailedOperation.DbQueryFailed	数据库查询错误。
FailedOperation.DbRecordCreateFailed	数据库创建失败。
FailedOperation.DbRecordUpdateFailed	数据库更新失败。
FailedOperation.ESQueryError	ES查询错误。
FailedOperation.ErrPrePaidResourceExpire	任务绑定的预付费套餐已过期。
FailedOperation.NoValidNodes	无有效节点。
FailedOperation.OrderOutOfCredit	账单欠费。
FailedOperation.PreResourceIDFailed	预付费资源id绑定错误。
FailedOperation.ResourceNotFound	资源不存在。
FailedOperation.SendRequest	发送授权请求失败。
FailedOperation.TagRequiredVerifyFailed	标签验证错误。
FailedOperation.TaskNotRunning	任务未运行。
FailedOperation.TaskNotSuspended	任务未暂停。
FailedOperation.TaskOperationNotAllowed	任务状态不允许当前操作。
FailedOperation.TaskTypeNotSame	批量拨测任务的类型不相同。
FailedOperation.TrialTaskExceed	试用任务量超时。

错误码	说明
FailedOperation.UnmarshalResponse	json解析失败。
FailedOperation.UserNoQcloudTAGFullAccess	鉴权失败。