自动化助手

API 文档

腾讯云

文档目录

API 文档
更新历史

最近更新时间：2025-03-10 01:57:42

第 25 次发布

发布时间：2025-03-10 01:57:30

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- RegisterInstanceInfo
  - 新增成员：Tags

第 24 次发布

发布时间：2024-09-13 02:06:44

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- Command
  - 新增成员：Scenes

第 23 次发布

发布时间：2024-08-14 02:23:52

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DescribeScenes

新增数据结构：

- Scene

第 22 次发布

发布时间：2023-12-15 20:16:37

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- AutomationAgentInfo
  - **修改成员：** SupportFeatures
- CommandDocument
  - **修改成员：** OutputCOSBucketUrl, OutputCOSKeyPrefix
- RegisterCodeInfo
  - **修改成员：** RegisterCodeId, Description, InstanceNamePrefix, RegisterLimit, ExpiredTime, IpAddressRange, Enabled, RegisteredCount, CreatedTime, UpdatedTime

第 21 次发布

发布时间：2023-10-11 02:22:08

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DeleteCommands
- DescribeQuotas

新增数据结构：

- GeneralResourceQuotaSet

第 20 次发布

发布时间：2023-09-18 02:20:44

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- CreateCommand
  - 新增入参：DefaultParameterConfs
- ModifyCommand
  - 新增入参：DefaultParameterConfs
- RunCommand
  - 新增入参：DefaultParameterConfs

第 19 次发布

发布时间：2023-08-24 01:26:36

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- CreateRegisterCode
- DeleteRegisterCodes
- DeleteRegisterInstance
- DescribeRegisterCodes
- DescribeRegisterInstances
- DisableRegisterCodes
- ModifyRegisterInstance

新增数据结构：

- DefaultParameterConf
- RegisterCodeInfo
- RegisterInstanceInfo

修改数据结构：

- Command
  - 新增成员：DefaultParameterConfs

第 18 次发布

发布时间：2023-02-20 01:49:40

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- AutomationAgentInfo
  - 新增成员：SupportFeatures
- CommandDocument
  - 新增成员：OutputCOSBucketUrl, OutputCOSKeyPrefix

第 17 次发布

发布时间：2021-12-09 08:10:32

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- CancelInvocation

第 16 次发布

发布时间：2021-12-03 08:11:00

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- Command
  - 新增成员：OutputCOSBucketUrl, OutputCOSKeyPrefix
- Invocation
  - 新增成员：OutputCOSBucketUrl, OutputCOSKeyPrefix
- TaskResult
  - 新增成员：OutputUrl, OutputUploadCOSErrorInfo

第 15 次发布

发布时间：2021-12-02 08:11:32

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- InvokeCommand
  - 新增入参：OutputCOSBucketUrl, OutputCOSKeyPrefix
- ModifyCommand
  - 新增入参：OutputCOSBucketUrl, OutputCOSKeyPrefix
- RunCommand
  - 新增入参：OutputCOSBucketUrl, OutputCOSKeyPrefix

第 14 次发布

发布时间：2021-11-24 08:13:01

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- CreateCommand
  - 新增入参：OutputCOSBucketUrl, OutputCOSKeyPrefix

第 13 次发布

发布时间：2021-09-23 08:12:30

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- CreateInvoker
- DeleteInvoker
- DescribeInvokerRecords
- DescribeInvokers
- DisableInvoker
- EnableInvoker
- ModifyInvoker

新增数据结构：

- Invoker
- InvokerRecord
- ScheduleSettings

第 12 次发布

发布时间：2021-09-01 08:11:51

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- Invocation
  - 新增成员：CommandContent, CommandType, Timeout, WorkingDirectory

第 11 次发布

发布时间：2021-07-28 08:12:03

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- Invocation
  - 新增成员：InvocationSource
- InvocationTask
  - 新增成员：InvocationSource

第 10 次发布

发布时间：2021-07-21 08:11:56

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- InvokeCommand
  - 新增入参：WorkingDirectory, Timeout

第 9 次发布

发布时间：2021-07-19 08:11:15

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- InvocationTask
  - 新增成员：ErrorInfo

第 8 次发布

发布时间：2021-07-14 08:12:04

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- CreateCommand
  - 新增入参：Username
- InvokeCommand
  - 新增入参：Username
- ModifyCommand
  - 新增入参：Username
- RunCommand
  - 新增入参：Username

修改数据结构：

- Command
  - 新增成员：Username
- CommandDocument
  - 新增成员：Username
- Invocation
  - 新增成员：Username

第 7 次发布

发布时间：2021-07-07 10:06:48

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- Invocation
  - 新增成员：InstanceKind

第 6 次发布

发布时间：2021-05-31 08:11:21

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- CreateCommand
  - 新增入参：Tags
- RunCommand
  - 新增入参：Tags

第 5 次发布

发布时间：2021-05-28 08:11:06

本次发布包含了以下内容：

改善已有的文档。

新增数据结构：

- Tag

修改数据结构：

- Command
  - 新增成员：Tags

第 4 次发布

发布时间：2021-04-25 10:20:20

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- TaskResult
  - 新增成员：Dropped

第 3 次发布

发布时间：2021-04-13 08:11:56

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- Command
  - 新增成员：FormattedDescription, CreatedBy

第 2 次发布

发布时间：2021-03-23 08:12:09

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- PreviewReplacedCommandContent

修改接口：

- CreateCommand
  - 新增入参：EnableParameter, DefaultParameters
- InvokeCommand
  - 新增入参：Parameters
- ModifyCommand
  - 新增入参：DefaultParameters
- RunCommand
  - 新增入参：EnableParameter, DefaultParameters, Parameters

修改数据结构：

- Command
  - 新增成员：EnableParameter, DefaultParameters
- Invocation
  - 新增成员：Parameters, DefaultParameters

第 1 次发布

发布时间：2021-02-02 08:11:53

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- CreateCommand
- DeleteCommand
- DescribeAutomationAgentStatus
- DescribeCommands
- DescribeInvocationTasks
- DescribeInvocations
- DescribeRegions
- InvokeCommand
- ModifyCommand
- RunCommand

新增数据结构：

- AutomationAgentInfo
- Command
- CommandDocument
- Filter
- Invocation
- InvocationTask
- InvocationTaskBasicInfo
- RegionInfo
- TaskResult

简介

最近更新时间：2025-04-25 01:54:43

高效安全的云服务器原生运维部署工具。

API 概览

最近更新时间：2025-01-20 01:24:21

远程命令相关接口

接口名称	接口功能	频率限制（次/秒）
CreateCommand	创建命令	20
ModifyCommand	修改命令	20
DeleteCommand	删除命令	20
DeleteCommands	批量删除命令	20
DescribeCommands	查询命令详情	20
PreviewReplacedCommandContent	命令预览	20
InvokeCommand	触发命令	20
RunCommand	执行命令	20
CancelInvocation	取消命令执行	20
DescribeInvocations	查询执行活动	20
DescribeInvocationTasks	查询执行任务	20
DescribeAutomationAgentStatus	查询客户端状态	60

定时执行相关接口

接口名称	接口功能	频率限制（次/秒）
CreateInvoker	创建执行器	20
ModifyInvoker	修改执行器	20
DeleteInvoker	删除执行器	20
EnableInvoker	启用执行器	20
DisableInvoker	停用执行器	20
DescribeInvokers	查询执行器	20
DescribeInvokerRecords	查询执行器执行记录	20

托管实例相关接口

接口名称	接口功能	频率限制（次/秒）
CreateRegisterCode	创建注册码	20
DisableRegisterCodes	批量禁用注册码	20
DeleteRegisterCodes	批量删除注册码	20
DescribeRegisterCodes	查询注册码	20
ModifyRegisterInstance	修改托管实例	20
DeleteRegisterInstance	删除托管实例	20
DescribeRegisterInstances	查询托管实例	20

场景相关接口

接口名称	接口功能	频率限制（次/秒）
DescribeScenes	查询场景	20

统计相关接口

接口名称	接口功能	频率限制（次/秒）
DescribeRegions	查询地域列表	20
DescribeQuotas	获取配额信息	20

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

调用方式
请求结构

最近更新时间：2025-05-21 01:29:50

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

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

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

2. 通信协议

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

3. 请求方法

支持的 HTTP 请求方法:

- POST（推荐）
- GET

POST 请求支持的 Content-Type 类型:

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

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

4. 字符编码

均使用 `UTF-8` 编码。

公共参数

最近更新时间：2025-03-26 02:03:56

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

公共参数的具体内容会因您使用的签名方法版本不同而有所差异。

使用签名方法 v3 的公共参数

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

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

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

参数名称	类型	必选	描述
Action	String	是	HTTP 请求头：X-TC-Action。操作的接口名称。取值参考接口文档输入参数章节关于公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。
Region	String	-	HTTP 请求头：X-TC-Region。地域参数，用来标识希望操作哪个地域的数据。取值参考接口文档中输入参数章节关于公共参数 Region 的说明。**注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。**
Timestamp	Integer	是	HTTP 请求头：X-TC-Timestamp。当前 UNIX 时间戳，可记录发起 API 请求的时间。例如 1529223702。**注意：如果与服务器时间相差超过5分钟，会引起签名过期错误。**
Version	String	是	HTTP 请求头：X-TC-Version。操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
Authorization	String	是	HTTP 标准身份认证头部字段，例如： TC3-HMAC-SHA256 Credential=AKID***/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024 其中， - TC3-HMAC-SHA256：签名方法，目前固定取该值； - Credential：签名凭证，AKID*** 是 SecretId；Date 是 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致；service 为具体产品名，通常为域名前缀。例如，域名 cvm.tencentcloudapi.com 意味着产品名是 cvm。本产品取值为 tat；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。

参数名称	类型	必选	描述
Version	String	是	操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
SignatureMethod	String	否	签名方式，目前支持 HmacSHA256 和 HmacSHA1。只有指定此参数为 HmacSHA256 时，才使用 HmacSHA256 算法验证签名，其他情况均使用 HmacSHA1 验证签名。
Token	String	否	即 安全凭证服务 所颁发的临时安全凭证中的 Token，使用时需要将 SecretId 和 SecretKey 的值替换为临时安全凭证中的 TmpSecretId 和 TmpSecretKey。使用长期密钥时不能设置此 Token 字段。
Language	String	否	指定接口返回的语言，仅部分接口支持此参数。取值：zh-CN，en-US。zh-CN 返回中文，en-US 返回英文。

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

HTTP GET 请求结构示例：

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

Host: cvm.tencentcloudapi.com
```

HTTP POST 请求结构示例：

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

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

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

地域列表

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

地域	取值
亚太东南（曼谷）	ap-bangkok
华北地区（北京）	ap-beijing
西南地区（成都）	ap-chengdu
西南地区（重庆）	ap-chongqing
华南地区（广州）	ap-guangzhou
港澳台地区（中国香港）	ap-hongkong
亚太东南（雅加达）	ap-jakarta
华东地区（南京）	ap-nanjing
亚太东北（首尔）	ap-seoul
华东地区（上海）	ap-shanghai
华东地区（上海金融）	ap-shanghai-fsi
华南地区（深圳金融）	ap-shenzhen-fsi
亚太东南（新加坡）	ap-singapore

地域	取值
亚太东北（东京）	ap-tokyo
欧洲地区（法兰克福）	eu-frankfurt
美国东部（弗吉尼亚）	na-ashburn
美国西部（硅谷）	na-siliconvalley
南美地区（圣保罗）	sa-saopaulo

签名方法 v3

最近更新时间：2024-12-25 01:59:31

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

推荐使用 API Explorer　　点击调试

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

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

为什么要进行签名

签名通过以下方式帮助保护请求：

1. 验证请求者的身份
   签名确保请求是由持有有效访问密钥的人发送的。请参阅控制台 云 API 密钥 页面获取密钥相关信息。

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

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

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

申请安全凭证

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

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

申请安全凭证的具体步骤如下：

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

签名版本 v3 签名过程

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

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

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

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

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

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

下面详细解释签名计算过程。

1. 拼接规范请求串

按如下伪代码格式拼接规范请求串（CanonicalRequest）：

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

字段名称	解释
HTTPRequestMethod	HTTP 请求方法（GET、POST）。此示例取值为 POST。
CanonicalURI	URI 参数，API 3.0 固定为正斜杠（/）。
CanonicalQueryString	发起 HTTP 请求 URL 中的查询字符串，对于 POST 请求，固定为空字符串""，对于 GET 请求，则为 URL 中问号（?）后面的字符串内容，例如：Limit=10&Offset=0。 注意：CanonicalQueryString 需要参考 RFC3986 进行 URLEncode 编码（特殊字符编码后需大写字母），字符集 UTF-8。推荐使用编程语言标准库进行编码。
CanonicalHeaders	参与签名的头部信息，至少包含 host 和 content-type 两个头部，也可加入其他头部参与签名以提高自身请求的唯一性和安全性，此示例额外增加了接口名头部。 拼接规则： 1. 头部 key 和 value 统一转成小写，并去掉首尾空格，按照 key:value\n 格式拼接； 2. 多个头部，按照头部 key（小写）的 ASCII 升序进行拼接。 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` 是二进制的数据，可能包含不可打印字符，将其转为十六进制字符串打印的输出分别为：
da98fb70dcf6b112dc21038d1eeeb3a95c74b4dcb12c1131f864f6066bd02be0，
8d70cbefb03939f929db64d32dc2ba89b1095620119fe3e050e2b18c5bd2752f，
b596b923aad85185e2d1f6659d2a062e0a86731226e021e61bfe06f7ed05f5af。

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

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

2）计算签名，伪代码如下：

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

此示例计算结果是 `10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f`。

4. 拼接 Authorization

按如下格式拼接 Authorization：

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

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

根据以上规则，示例中得到的值为：

```
TC3-HMAC-SHA256 Credential=AKID********************************/2019-02-25/cvm/tc3_request, SignedHeaders=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

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

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

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

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

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

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

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

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

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

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

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

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

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

Python

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

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

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

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

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

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


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

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

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

Golang

```
package main

import (
"crypto/hmac"
```

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

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

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

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

// step 1: build canonical request string
httpRequestMethod := "POST"
canonicalURI := "/"
canonicalQueryString := ""
canonicalHeaders := fmt.Sprintf("content-type:%s\nhost:%s\nx-tc-action:%s\n",
"application/json; charset=utf-8", host, strings.ToLower(action))
signedHeaders := "content-type;host;x-tc-action"
payload := `{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}`
hashedRequestPayload := sha256hex(payload)
canonicalRequest := fmt.Sprintf("%s\n%s\n%s\n%s\n%s\n%s",
httpRequestMethod,
canonicalURI,
canonicalQueryString,
canonicalHeaders,
signedHeaders,
hashedRequestPayload)
fmt.Println(canonicalRequest)

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

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

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

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

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

PHP

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

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

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

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

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

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

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

Ruby

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

```ruby
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=#{si
gnature}"
puts authorization

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

DotNet

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

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

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

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

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

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

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

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

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

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

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

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

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

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

NodeJS

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

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

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

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

function main(){
```

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

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

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

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

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

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

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

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

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

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

C++

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

using namespace std;

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

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

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

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

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

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

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

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

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

return (ss.str());
}

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

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

```
}
return output;
}

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

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

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

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

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

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

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

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

C

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

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

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

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

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

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


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

}

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

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

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

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

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

// ************* 步骤 1: 拼接规范请求串 *************
const char* http_request_method = "POST";
const char* canonical_uri = "/";
const char* canonical_query_string = "";
char canonical_headers[100] = {"content-type:application/json; charset=utf-8\nhost:"};
strcat(canonical_headers, host);
strcat(canonical_headers, "\nx-tc-action:");
char value[100] = {0};
lowercase(action, value);
strcat(canonical_headers, value);
strcat(canonical_headers, "\n");
const char* signed_headers = "content-type;host;x-tc-action";
const char* payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"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 01:59:31

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

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

推荐使用 API Explorer　　点击调试

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

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

1. 申请安全凭证

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

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

申请安全凭证的具体步骤如下：

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

注意：每个账号最多可以拥有两对 SecretId/SecretKey。

2. 生成签名串

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

假设用户的 SecretId 和 SecretKey 分别是：

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

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

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

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

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

2.1. 对参数排序

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

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

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

2.2. 拼接请求字符串

此步骤生成请求字符串。
将把上一步排序好的请求参数格式化成“参数名称=参数值”的形式，如对 Action 参数，其参数名称为 "Action" ，参数值为 "DescribeInstances" ，因此格式化后就为 Action=DescribeInstances 。
注意：“参数值”为原始值而非 url 编码后的值。

然后将格式化后的各个参数用"&"拼接在一起，最终生成的请求字符串为:

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

2.3. 拼接签名原文字符串

此步骤生成签名原文字符串。
签名原文字符串由以下几个参数构成:

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

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

示例的拼接结果为:

```
GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-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 编码前需要先以 UTF-8 进行编码。

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

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

4. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理。

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

5. 签名演示

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

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

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

- Signature Demo

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

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

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

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

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

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

Java

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

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

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

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

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

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

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

Python

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

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

import requests

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

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

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

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

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

Golang

```
package main

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

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

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

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

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

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

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

PHP

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

ksort($param);

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

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

Ruby

```
# -*- 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 reques
tHost, 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 sigInParam = MakeSignPlainText(headers, "GET", endpoint, "/");
string sigOutParam = Sign(SECRET_KEY, sigInParam);
Console.WriteLine(sigOutParam);
}
}
```

NodeJS

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

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

```
function formatSignString(reqMethod, endpoint, path, strParam){
let strSign = reqMethod + endpoint + path + "?" + strParam.slice(1);
return strSign;
}
function sha1(secretKey, strsign){
let signMethodMap = {'HmacSHA1': "sha1"};
let hmac = crypto.createHmac(signMethodMap['HmacSHA1'], secretKey || "");
return hmac.update(Buffer.from(strsign, 'utf8')).digest('base64')
}

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

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

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

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

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

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

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

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

返回结果

最近更新时间：2024-03-12 01:48:54

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

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

正确返回结果

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

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

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

错误返回结果

若调用失败，其返回值示例如下为：

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

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

公共错误码

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

参数类型

最近更新时间：2022-08-10 06:47:44

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。

远程命令相关接口
创建命令

最近更新时间：2025-04-25 01:54:40

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于创建命令。

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：CreateCommand。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
CommandName	是	String	命令名称。名称仅支持中文、英文、数字、下划线、分隔符"-"、小数点，最大长度不能超60个字节。 示例值：tat-command
Content	是	String	Base64编码后的命令内容，长度不可超过64KB。 示例值：cHdk
Description	否	String	命令描述。不超过120字符。 示例值：this is a command
CommandType	否	String	命令类型，目前支持取值：SHELL、POWERSHELL、BAT。默认：SHELL。 示例值：SHELL
WorkingDirectory	否	String	命令执行路径，对于 SHELL 命令默认为 /root，对于 POWERSHELL 命令默认为 C:\Program Files\qcloud\tat_agent\workdir。 示例值：/root
Timeout	否	Integer	命令超时时间，默认60秒。取值范围[1, 86400]。 示例值：60
EnableParameter	否	Boolean	是否启用自定义参数功能。 一旦创建，此值不提供修改。 默认值：false。 示例值：false
DefaultParameters	否	String	启用自定义参数功能时，自定义参数的默认取值。字段类型为json encoded string。 如：{"varA": "222"}。 key为自定义参数名称，value为该参数的默认取值。kv均为字符串型。 如果InvokeCommand时未提供参数取值，将使用这里的默认值进行替换。 参数不支持同时指定 DefaultParameters 和 DefaultParameterConfs 。 仅在 EnableParameter 参数为 true 时，才允许设置此参数。 自定义参数最多20个。 自定义参数名称需符合以下规范：字符数目上限64，可选范围【a-zA-Z0-9-_】。 示例值：{"varA": "222"}
DefaultParameterConfs.N	否	Array of DefaultParameterConf	自定义参数数组。 如果InvokeCommand时未提供参数取值，将使用这里的默认值进行替换。 参数不支持同时指定 DefaultParameters 和 DefaultParameterConfs 。

参数名称	必选	类型	描述
			仅在 EnableParameter 参数为 true 时，才允许设置此参数。 自定义参数最多20个。 示例值：[{ "ParameterName": "test01", "ParameterValue": "12345", "ParameterDescription": "for test01" }, { "ParameterName": "test02", "ParameterValue": "12345", "ParameterDescription": "for test02"}]
Tags.N	否	Array of Tag	为命令关联的标签，列表长度不超过10。 示例值：[{ "Key": "test", "Value": "test" }]
Username	否	String	在 CVM 或 Lighthouse 实例中执行命令的用户名称。 使用最小权限执行命令是权限管理的最佳实践，建议您以普通用户身份执行云助手命令。 默认情况下，在 Linux 实例中以 root 用户执行命令；在Windows 实例中以 System 用户执行命令。 示例值：root
OutputCOSBucketUrl	否	String	指定日志上传的cos bucket 地址，必须以https开头，如 https://BucketName-123454321.cos.ap-beijing.myqcloud.com。 示例值：https://BucketName-123454321.cos.ap-beijing.myqcloud.com
OutputCOSKeyPrefix	否	String	指定日志在cos bucket中的目录，目录命名有如下规则： 1. 可用数字、中英文和可见字符的组合，长度最多为60。 2. 用 / 分割路径，可快速创建子目录。 3. 不允许连续 / ；不允许以 / 开头；不允许以..作为文件夹名称 示例值：aa/bb/cc

3. 输出参数

参数名称	类型	描述
CommandId	String	命令ID。 示例值：cmd-543xxk0d
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 创建命令

输入示例

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

{
"CommandName": "hello-command",
"Description": "hello world",
"Content": "bHM=",
"CommandType": "SHELL",
"WorkingDirectory": "/tmp",
"Timeout": 60
}
```

输出示例

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

```
"RequestId": "9bea970a-0bab-495f-b0dd-de5eedfdf611",
"CommandId": "cmd-7efujjs6"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.CommandContentInvalid	Command 内容无效。
InvalidParameterValue.CommandNameDuplicated	Command 名称重复。
InvalidParameterValue.InvalidCommandName	Command 名称无效。
InvalidParameterValue.InvalidContent	命令内容无效。
InvalidParameterValue.InvalidOutputCOSBucketUrl	OutputCOSBucketUrl 无效。
InvalidParameterValue.InvalidOutputCOSKeyPrefix	OutputCOSKeyPrefix 无效。
InvalidParameterValue.InvalidUsername	用户名不合法。
InvalidParameterValue.InvalidWorkingDirectory	命令执行路径不合法。
InvalidParameterValue.ParameterDisabled	未启用自定义参数功能。
InvalidParameterValue.ParameterInvalidJsonFormat	参数为非法 json string 格式。
InvalidParameterValue.ParameterKeyContainsInvalidChar	参数 Key 包含非法字符。
InvalidParameterValue.ParameterKeyDuplicated	参数 Key 重复。
InvalidParameterValue.ParameterKeyLenExceeded	参数 Key 过长。

错误码	描述
InvalidParameterValue.ParameterNumberExceeded	参数数目过多。
InvalidParameterValue.ParameterValueNotString	参数 Value 非 string 类型。
InvalidParameterValue.Range	参数取值范围不合法。
InvalidParameterValue.SupportParametersOnlyIfEnableParameter	未启用自定义参数功能。
InvalidParameterValue.TooLong	长度超过限制。
MissingParameter	缺少参数错误。
ResourceUnavailable.UserHasNoQuotaCode	用户配额已用完
UnauthorizedOperation.CamAuthFailed	CAM鉴权失败。
UnauthorizedOperation.InvalidToken	Token 无效。
UnauthorizedOperation.MFAExpired	Multi-Factor Authentication(MFA) 过期。
UnauthorizedOperation.MFANotFound	Multi-Factor Authentication(MFA) 不存在。
UnsupportedOperation	操作不支持。

修改命令

最近更新时间：2025-04-25 01:54:39

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于修改命令。

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：ModifyCommand。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
CommandId	是	String	命令ID。可通过 DescribeCommands(查询命令详情) 接口获取。 示例值：cmd-7efujjs6
CommandName	否	String	命令名称。名称仅支持中文、英文、数字、下划线、分隔符"-"、小数点，最大长度不能超60个字节。 示例值：tat-command
Description	否	String	命令描述。不超过120字符。 示例值：this is a command
Content	否	String	Base64编码后的命令内容，长度不可超过64KB。 示例值：cHdk
CommandType	否	String	命令类型，目前支持取值：SHELL、POWERSHELL、BAT。 示例值：SHELL
WorkingDirectory	否	String	命令执行路径。 示例值：/root
Timeout	否	Integer	命令超时时间。取值范围[1, 86400]。 示例值：60
DefaultParameters	否	String	启用自定义参数功能时，自定义参数的默认取值。字段类型为json encoded string。 如：{"varA": "222"}。 参数不支持同时指定 DefaultParameters 和 DefaultParameterConfs 。 采取整体全覆盖式修改，即修改时必须提供所有新默认值。 仅在命令的 EnableParameter 为 true 时，才允许修改此参数。可通过 DescribeCommands(查询命令详情) 接口获取命令的 EnableParameter 设置。 key为自定义参数名称，value为该参数的默认取值。kv均为字符串型。 自定义参数最多20个。 自定义参数名称需符合以下规范：字符数目上限64，可选范围【a-zA-Z0-9-_】。 示例值：{"varA": "222"}
DefaultParameterConfs.N	否	Array of DefaultParameterConf	自定义参数数组。如果 InvokeCommand 时未提供参数取值，将使用这里的默认值进行替换。 参数不支持同时指定 DefaultParameters 和 DefaultParameterConfs 。 仅在命令的 EnableParameter 为 true 时，才允许修改此参数。可通过 DescribeCommands(查询命令详情) 接口获取命令的 EnableParameter 设置。 自定义参数最多20个。 示例值：[{ "ParameterName": "test01", "ParameterValue": "12345",

参数名称	必选	类型	描述
			"ParameterDescription": "for test01" }, { "ParameterName": "test02", "ParameterValue": "12345", "ParameterDescription": "for test02"}]
Username	否	String	在 CVM 或 Lighthouse 实例中执行命令的用户名称。 使用最小权限执行命令是权限管理的最佳实践，建议您以普通用户身份执行云助手命令。 示例值：root
OutputCOSBucketUrl	否	String	指定日志上传的cos bucket 地址，必须以https开头，如 https://BucketName-123454321.cos.ap-beijing.myqcloud.com。 示例值：https://BucketName-123454321.cos.ap-beijing.myqcloud.com
OutputCOSKeyPrefix	否	String	指定日志在cos bucket中的目录，目录命名有如下规则： 1. 可用数字、中英文和可见字符的组合，长度最多为60。 2. 用 / 分割路径，可快速创建子目录。 3. 不允许连续 / ；不允许以 / 开头；不允许以..作为文件夹名称。 示例值：aa/bb/cc

3. 输出参数

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

4. 示例

示例1 修改命令

输入示例

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

{
"CommandName": "second-command",
"Description": "hello world!",
"Content": "cHM=",
"CommandType": "SHELL",
"WorkingDirectory": "/root",
"Timeout": 600,
"CommandId": "cmd-63usjhmq"
}
```

输出示例

```
{
"Response": {
"RequestId": "9bea970a-0bab-495f-b0dd-de5eedfdf611"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.CommandContentInvalid	Command 内容无效。
InvalidParameterValue.CommandNameDuplicated	Command 名称重复。
InvalidParameterValue.InvalidCommandId	CommandId 无效。
InvalidParameterValue.InvalidCommandName	Command 名称无效。
InvalidParameterValue.InvalidOutputCOSBucketUrl	OutputCOSBucketUrl 无效。
InvalidParameterValue.InvalidOutputCOSKeyPrefix	OutputCOSKeyPrefix 无效。
InvalidParameterValue.InvalidWorkingDirectory	命令执行路径不合法。
InvalidParameterValue.ParameterInvalidJsonFormat	参数为非法 json string 格式。
InvalidParameterValue.ParameterKeyContainsInvalidChar	参数 Key 包含非法字符。
InvalidParameterValue.ParameterKeyDuplicated	参数 Key 重复。
InvalidParameterValue.ParameterKeyLenExceeded	参数 Key 过长。
InvalidParameterValue.ParameterNumberExceeded	参数数目过多。
InvalidParameterValue.ParameterValueNotString	参数 Value 非 string 类型。
InvalidParameterValue.Range	参数取值范围不合法。
InvalidParameterValue.SupportParametersOnlyIfEnableParameter	未启用自定义参数功能。
InvalidParameterValue.TooLong	长度超过限制。
ResourceNotFound.CommandNotFound	命令不存在。
UnauthorizedOperation.CamAuthFailed	CAM鉴权失败。

错误码	描述
UnauthorizedOperation.InvalidToken	Token 无效。
UnauthorizedOperation.MFAExpired	Multi-Factor Authentication(MFA) 过期。
UnauthorizedOperation.MFANotFound	Multi-Factor Authentication(MFA) 不存在。

删除命令

最近更新时间：2025-04-25 01:54:40

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于删除命令。
如果命令与执行器关联，则无法被删除。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DeleteCommand。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
CommandId	是	String	待删除的命令 ID。可通过 DescribeCommands(查询命令详情) 接口获取。 示例值：cmd-7efujjs6

3. 输出参数

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

4. 示例

示例1 删除命令

输入示例

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

{
"CommandId": "cmd-7efujjs6"
}
```

输出示例

```
{
"Response": {
"RequestId": "7f79b764-bc0f-471b-89c1-ca1b93cf7e8d"
```

```
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError	内部错误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.InvalidCommandId	CommandId 无效。
ResourceNotFound.CommandNotFound	命令不存在。
ResourceUnavailable.CommandInExecuting	命令正在执行中。
ResourceUnavailable.CommandInInvoker	命令已关联执行器。
UnauthorizedOperation.CamAuthFailed	CAM鉴权失败。
UnauthorizedOperation.InvalidToken	Token 无效。
UnauthorizedOperation.MFAExpired	Multi-Factor Authentication(MFA) 过期。
UnauthorizedOperation.MFANotFound	Multi-Factor Authentication(MFA) 不存在。

批量删除命令

最近更新时间：2025-04-25 01:54:40

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

批量删除命令接口

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：DeleteCommands。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
CommandIds.N	是	Array of String	待删除的命令 ID。可通过 DescribeCommands(查询命令详情) 接口获取。 示例值：["cmd-0v4tysy3","cmd-0v4tysy4"]

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 批量删除命令

输入示例

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

{
"CommandIds": [
"cmd-7efujjs6",
"cmd-7efujjs7"
]
}
```

输出示例

```
{
"Response": {
"RequestId": "7f79b764-bc0f-471b-89c1-ca1b93cf7e8d"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError	内部错误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.InvalidCommandId	CommandId 无效。
ResourceNotFound.CommandNotFound	命令不存在。
ResourceUnavailable.CommandInExecuting	命令正在执行中。
ResourceUnavailable.CommandInInvoker	命令已关联执行器。
UnauthorizedOperation.CamAuthFailed	CAM鉴权失败。
UnauthorizedOperation.InvalidToken	Token 无效。
UnauthorizedOperation.MFAExpired	Multi-Factor Authentication(MFA) 过期。
UnauthorizedOperation.MFANotFound	Multi-Factor Authentication(MFA) 不存在。

查询命令详情

最近更新时间：2025-04-25 01:54:40

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于查询命令详情。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeCommands。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
CommandIds.N	否	Array of String	命令ID列表，每次请求的上限为100。参数不支持同时指定 CommandIds 和 Filters 。 示例值：["cmd-2x8vxx0d"]
Filters.N	否	Array of Filter	过滤条件。 - command-id - String - 是否必填：否 -（过滤条件）按照命令ID过滤。 - command-name - String - 是否必填：否 -（过滤条件）按照命令名称过滤。 - command-type - String - 是否必填：否 -（过滤条件）按照命令类型过滤，取值为 SHELL、POWERSHELL、BAT。 - scene-id - String - 是否必填：否 -（过滤条件）按照场景ID过滤。可通过 DescribeScenes(查询场景) 接口获取场景ID。 - created-by - String - 是否必填：否 -（过滤条件）按照命令创建者过滤，取值为 TAT 或 USER。TAT 代表公共命令，USER 代表由用户创建的命令。 - tag-key - String - 是否必填：否 -（过滤条件）按照标签键进行过滤。 - tag-value - String - 是否必填：否 -（过滤条件）按照标签值进行过滤。 - tag:tag-key - String - 是否必填：否 -（过滤条件）按照标签键值对进行过滤。 tag-key使用具体的标签键进行替换。使用请参考示例4 每次请求的 Filters 的上限为10， Filter.Values 的上限为5。参数不支持同时指定 CommandIds 和 Filters 。 示例值：[{ "Name": "command-id", "Values": ["cmd-nrncl292"] }]
Limit	否	Integer	返回数量，默认为20，最大值为100。关于 Limit 的更进一步介绍请参考 API 简介中的相关小节。 示例值：10
Offset	否	Integer	偏移量，默认为0。关于 Offset 的更进一步介绍请参考 API 简介中的相关小节。 示例值：0

3. 输出参数

参数名称	类型	描述
TotalCount	Integer	符合条件的命令总数。 示例值：0
CommandSet	Array of Command	命令详情列表。

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 使用 CommandId 查询命令

输入示例

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

{
"CommandIds": [
"cmd-dvstpcyy"
],
"Offset": 0,
"Limit": 20
}
```

输出示例

```
{
"Response": {
"RequestId": "eb973a12-71e3-4c0c-b1d8-4b863e5f5daf",
"TotalCount": 1,
"CommandSet": [
{
"CommandId": "cmd-dvstpcyy",
"CommandName": "run-command",
"Description": "whoami",
"FormattedDescription": "",
"CreatedBy": "USER",
"Content": "d2hvYW1p",
"CommandType": "SHELL",
"WorkingDirectory": "/root/",
"Timeout": 60,
"EnableParameter": false,
"DefaultParameters": "{\"varA\": \"222\"}",
"DefaultParameterConfs": [],
"Scenes": [],
"Username": "root",
"Tags": [
{
"Value": "test-key",
"Key": "test-value"
}
],
"CreatedTime": "2020-11-02T02:48:11+00:00",
"UpdatedTime": "2020-11-02T02:48:11+00:00",
"OutputCOSBucketUrl": "https://BucketName-123454321.cos.ap-beijing.myqcloud.com",
"OutputCOSKeyPrefix": "logs"
```

```
}
]
}
}
```

示例2 使用 created-by Filter 查询

输入示例

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

{
"Offset": 0,
"Limit": 1,
"Filters": [
{
"Name": "created-by",
"Values": [
"USER"
]
}
]
}
```

输出示例

```
{
"Response": {
"RequestId": "6b215093-e1f6-4803-b84a-a230849e88d1",
"TotalCount": 2,
"CommandSet": [
{
"CommandId": "cmd-hb2q34lk",
"CommandName": "second-command",
"Description": "ps",
"FormattedDescription": "",
"DefaultParameterConfs": [],
"Scenes": [],
"CreatedBy": "USER",
"Content": "cHM=",
"CommandType": "SHELL",
"WorkingDirectory": "/root/",
"Timeout": 60,
"EnableParameter": true,
"DefaultParameters": "{\"varA\": \"222\"}",
"Username": "root",
"Tags": [
{
"Value": "test-key",
"Key": "test-value"
}
],
```

```
"CreatedTime": "2020-10-30T07:19:52+00:00",
"UpdatedTime": "2020-10-30T07:19:52+00:00",
"OutputCOSBucketUrl": "https://BucketName-123454321.cos.ap-beijing.myqcloud.com",
"OutputCOSKeyPrefix": "logs"
}
]
}
}
```


示例3 使用 command-name Filter 查询

输入示例

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

{
"Offset": 0,
"Limit": 20,
"Filters": [
{
"Name": "command-name",
"Values": [
"second-command",
"first-command"
]
}
]
}
```

输出示例


```
{
"Response": {
"RequestId": "6b215093-e1f6-4803-b84a-a230849e88d1",
"TotalCount": 2,
"CommandSet": [
{
"CommandId": "cmd-hb2q34lk",
"CommandName": "second-command",
"Description": "ps",
"FormattedDescription": "",
"DefaultParameterConfs": [],
"Scenes": [],
"CreatedBy": "USER",
"Content": "cHM=",
"CommandType": "SHELL",
"WorkingDirectory": "/root/",
"Timeout": 60,
"EnableParameter": false,
"DefaultParameters": "{\"varA\": \"222\"}",
"Username": "root",
"Tags": [
```

```
{
"Value": "test-key",
"Key": "test-value"
}
],
"CreatedTime": "2020-10-30T07:19:52+00:00",
"UpdatedTime": "2020-10-30T07:19:52+00:00",
"OutputCOSBucketUrl": "https://BucketName-123454321.cos.ap-beijing.myqcloud.com",
"OutputCOSKeyPrefix": "aa/bb/cc"
},
{
"CommandId": "cmd-63usjhmq",
"CommandName": "first-command",
"Description": "hello world!",
"FormattedDescription": "",
"DefaultParameterConfs": [],
"Scenes": [],
"CreatedBy": "USER",
"Content": "cHM=",
"CommandType": "SHELL",
"WorkingDirectory": "/root",
"Timeout": 600,
"EnableParameter": false,
"DefaultParameters": "{\"varA\": \"222\"}",
"Username": "root",
"Tags": [
{
"Value": "test-key",
"Key": "test-value"
}
],
"CreatedTime": "2020-10-26T11:26:07+00:00",
"UpdatedTime": "2020-11-09T08:12:45+00:00",
"OutputCOSBucketUrl": "https://BucketName-123454321.cos.ap-beijing.myqcloud.com",
"OutputCOSKeyPrefix": "logs"
}
]
}
}
```


示例4 使用tag:tag-key Filter 查询命令

输入示例

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

{
"Filters": [
{
"Values": [
"test-value"
],
```

```
"Name": "tag:test-key"
}
]
}
```

输出示例

```
{
"Response": {
"RequestId": "33d3d954-f73a-4a7f-869b-8792bc7a6f13",
"TotalCount": 1,
"CommandSet": [
{
"CommandId": "cmd-38ps9q4p",
"CommandName": "tag-test-1",
"Description": "",
"FormattedDescription": "",
"DefaultParameterConfs": [],
"Scenes": [],
"CreatedBy": "USER",
"Content": "cHMK",
"CommandType": "SHELL",
"WorkingDirectory": "/root",
"Timeout": 60,
"EnableParameter": false,
"DefaultParameters": "{\"varA\": \"222\"}",
"Username": "root",
"Tags": [
{
"Key": "test-key",
"Value": "test-value"
}
],
"CreatedTime": "2021-05-12T02:49:04Z",
"UpdatedTime": "2021-05-12T02:49:04Z",
"OutputCOSBucketUrl": "https://BucketName-123454321.cos.ap-beijing.myqcloud.com",
"OutputCOSKeyPrefix": "images"
}
]
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameter.ConflictParameter	参数冲突。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.InvalidCommandId	CommandId 无效。
InvalidParameterValue.InvalidFilter	Filter 无效。
UnauthorizedOperation.CamAuthFailed	CAM鉴权失败。
UnauthorizedOperation.InvalidToken	Token 无效。
UnauthorizedOperation.MFAExpired	Multi-Factor Authentication(MFA) 过期。
UnauthorizedOperation.MFANotFound	Multi-Factor Authentication(MFA) 不存在。

命令预览

最近更新时间：2025-04-25 01:54:39

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于预览自定义参数替换后的命令内容。不会触发真实执行。

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：PreviewReplacedCommandContent。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
Parameters	否	String	本次预览采用的自定义参数。字段类型为 json encoded string，如：{"varA": "222"}。 仅在命令的 EnableParameter 为 true 时，才允许设置此参数。可通过 DescribeCommands(查询命令详情) 接口获取命令的 EnableParameter 设置。 如果有设置过 DefaultParameters 或 DefaultParameterConfs，会与 Parameters 进行叠加，优先使用 Parameters 的值。 key 为自定义参数名称，value 为该参数的取值。kv 均为字符串型。 自定义参数最多 20 个。 自定义参数名称需符合以下规范：字符数目上限 64，可选范围【a-zA-Z0-9-_】。 如果将预览的 CommandId 设置过 DefaultParameters，本参数可以为空。 示例值：{"varA": "222"}
CommandId	否	String	要进行替换预览的命令。 可通过 DescribeCommands(查询命令详情) 接口获取。 CommandId 与 Content，必须且只能提供一个。 示例值：cmd-abcdefgh
Content	否	String	要预览的命令内容，经 Base64 编码，长度不可超过 64KB。 CommandId 与 Content，必须且只能提供一个。 示例值：bHMge3thfX0KZWNobyB7e2J9fQ==

3. 输出参数

参数名称	类型	描述
ReplacedContent	String	自定义参数替换后的，经Base64编码的命令内容。 示例值：bHMge3thfX0KZWNobyB7e2J9fQ==
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 预览content替换

输入示例

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

{
"Parameters": "{\"a\": \"123\"}",
"Content": "bHMge3thfX0KZWNobyB7e2J9fSB7e2N9fQ=="
}
```

输出示例

```
{
"Response": {
"RequestId": "0b4c6010-42a7-45cd-b8c3-daa7de930e82",
"ReplacedContent": "bHMgMTIzCmVjaG8ge3tifX0ge3tjfX0="
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter.ConflictParameter	参数冲突。
InvalidParameter.ParameterNameDuplicated	参数名称重复。
InvalidParameterValue.CommandContentInvalid	Command 内容无效。
InvalidParameterValue.CommandNameDuplicated	Command 名称重复。
InvalidParameterValue.InvalidCommandId	CommandId 无效。

错误码	描述
InvalidParameterValue.LackOfParameterInfo	已启用自定义参数功能，但缺失自定义参数信息。
InvalidParameterValue.LackOfParameters	未提供 Parameters 信息。
InvalidParameterValue.ParameterDisabled	未启用自定义参数功能。
InvalidParameterValue.ParameterInvalidJsonFormat	参数为非法 json string 格式。
InvalidParameterValue.ParameterKeyContainsInvalidChar	参数 Key 包含非法字符。
InvalidParameterValue.ParameterKeyDuplicated	参数 Key 重复。
InvalidParameterValue.ParameterKeyLenExceeded	参数 Key 过长。
InvalidParameterValue.ParameterNumberExceeded	参数数目过多。
InvalidParameterValue.ParameterValueNotString	参数 Value 非 string 类型。
InvalidParameterValue.Range	参数取值范围不合法。
InvalidParameterValue.SupportParametersOnlyIfEnableParameter	未启用自定义参数功能。
ResourceNotFound.CommandNotFound	命令不存在。
UnauthorizedOperation.CamAuthFailed	CAM鉴权失败。
UnauthorizedOperation.InvalidToken	Token 无效。
UnauthorizedOperation.MFAExpired	Multi-Factor Authentication(MFA) 过期。
UnauthorizedOperation.MFANotFound	Multi-Factor Authentication(MFA) 不存在。

触发命令

最近更新时间：2025-06-11 01:53:10

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

在指定的实例上触发命令，调用成功返回执行活动ID（inv-xxxxxxxx），每个执行活动内部有一个或多个执行任务（invt-xxxxxxxx），每个执行任务代表命令在一台 CVM 或一台 Lighthouse 上的执行记录。

- 如果指定实例未安装 agent，或 agent 不在线，返回失败
- 如果命令类型与 agent 运行环境不符，返回失败
- 指定的实例需要处于 VPC 网络
- 指定的实例需要处于 RUNNING 状态
- 不可同时指定 CVM 和 Lighthouse

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：InvokeCommand。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
CommandId	是	String	待触发的命令ID。可通过 DescribeCommands(查询命令详情) 接口获取。 示例值：cmd-ffxdx79i
InstanceIds.N	是	Array of String	待执行命令的实例ID列表，上限200。 可通过对应云产品的查询实例接口获取实例 ID。目前支持实例类型： - CVM - Lighthouse - TAT 托管实例 示例值：["lhins-ar5wyn40"]
Parameters	否	String	Command 的自定义参数。字段类型为json encoded string。如：{"varA": "222"}。 key为自定义参数名称，value为该参数的默认取值。kv均为字符串型。 仅在命令的 EnableParameter 为 true 时，才允许设置此参数。可通过 DescribeCommands(查询命令详情) 接口获取命令的 EnableParameter 设置。 如果未提供该参数取值，将使用 Command 的 DefaultParameters 或 DefaultParameterConfs 进行替换。 自定义参数最多20个。 自定义参数名称需符合以下规范：字符数目上限64，可选范围【a-zA-Z0-9-_】。 示例值：{"varA": "222"}
Username	否	String	在 CVM 或 Lighthouse 实例中执行命令的用户名称。 使用最小权限执行命令是权限管理的最佳实践，建议您以普通用户身份执行云助手命令。若不填，默认以 Command 配置的 Username 执行。 示例值：root
WorkingDirectory	否	String	命令执行路径，默认以Command配置的WorkingDirectory执行。 示例值：/root
Timeout	否	Integer	命令超时时间，取值范围[1, 86400]。默认以Command配置的Timeout执行。 示例值：60

参数名称	必选	类型	描述
OutputCOSBucketUrl	否	String	指定日志上传的cos bucket 地址，必须以https开头，如 https://BucketName-123454321.cos.ap-beijing.myqcloud.com。 示例值：https://BucketName-123454321.cos.ap-beijing.myqcloud.com
OutputCOSKeyPrefix	否	String	指定日志在cos bucket中的目录，目录命名有如下规则： 1. 可用数字、中英文和可见字符的组合，长度最多为60。 2. 用 / 分割路径，可快速创建子目录。 3. 不允许连续 / ；不允许以 / 开头；不允许以..作为文件夹名称。 示例值：aa/bb/cc

3. 输出参数

参数名称	类型	描述
InvocationId	String	执行活动ID。 示例值：inv-8xgjrytm
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 在 Lighthouse 触发命令

输入示例

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

{
"CommandId": "cmd-ffxdx79i",
"InstanceIds": [
"lhins-ar5wyn40"
]
}
```

输出示例

```
{
"Response": {
"RequestId": "41417f50-51b5-4c8d-85b7-f6c508cb228f",
"InvocationId": "inv-8xgjrytm"
}
}
```

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.CVMError	调用 CVM 失败。
FailedOperation.LighthouseError	调用 Lighthouse 失败。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameter.InvalidUsername	无效用户名。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.AgentUnsupportedCommandType	Agent不支持此命令类型。
InvalidParameterValue.InconsistentInstance	实例类型不一致。
InvalidParameterValue.InvalidCommandId	CommandId 无效。
InvalidParameterValue.InvalidInstanceId	实例ID无效。
InvalidParameterValue.InvalidOutputCOSBucketUrl	OutputCOSBucketUrl 无效。
InvalidParameterValue.InvalidOutputCOSKeyPrefix	OutputCOSKeyPrefix 无效。
InvalidParameterValue.InvalidWorkingDirectory	命令执行路径不合法。
InvalidParameterValue.LackOfParameterInfo	已启用自定义参数功能，但缺失自定义参数信息。
InvalidParameterValue.LimitExceeded	超过参数限制。
InvalidParameterValue.ParameterDisabled	未启用自定义参数功能。
InvalidParameterValue.ParameterInvalidJsonFormat	参数为非法 json string 格式。
InvalidParameterValue.ParameterKeyContainsInvalidChar	参数 Key 包含非法字符。
InvalidParameterValue.ParameterKeyDuplicated	参数 Key 重复。
InvalidParameterValue.ParameterKeyLenExceeded	参数 Key 过长。
InvalidParameterValue.ParameterNumberExceeded	参数数目过多。
InvalidParameterValue.ParameterValueNotString	参数 Value 非 string 类型。
InvalidParameterValue.SupportParametersOnlyIfEnableParameter	未启用自定义参数功能。
OperationDenied	操作被拒绝。

错误码	描述
ResourceNotFound.CommandNotFound	命令不存在。
ResourceNotFound.InstanceNotFound	实例不存在。
ResourceNotFound.RoleNotFound	角色不存在。
ResourceUnavailable	资源不可用。
ResourceUnavailable.AgentNotInstalled	Agent 未安装。
ResourceUnavailable.AgentStatusNotOnline	Agent 不在线。
ResourceUnavailable.InstanceStateNotRunning	实例未处于运行中。
ResourceUnavailable.LighthouseUnsupportedRegion	Lighthouse 尚不支持指定的地域。
UnauthorizedOperation.CamAuthFailed	CAM鉴权失败。
UnauthorizedOperation.InvalidToken	Token 无效。
UnauthorizedOperation.MFAExpired	Multi-Factor Authentication(MFA) 过期。
UnauthorizedOperation.MFANotFound	Multi-Factor Authentication(MFA) 不存在。

执行命令

最近更新时间：2025-06-11 01:53:09

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

执行命令，调用成功返回执行活动ID（inv-xxxxxxxx），每个执行活动内部有一个或多个执行任务（invt-xxxxxxxx），每个执行任务代表命令在一台 CVM 或一台 Lighthouse 上的执行记录。

- 如果指定实例未安装 agent，或 agent 不在线，返回失败
- 如果命令类型与 agent 运行环境不符，返回失败
- 指定的实例需要处于 VPC 网络
- 指定的实例需要处于 RUNNING 状态
- 不可同时指定 CVM 和 Lighthouse

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：RunCommand。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
Content	是	String	Base64编码后的命令内容，长度不可超过64KB。 示例值：cHdk
InstanceIds.N	是	Array of String	待执行命令的实例ID列表，上限200。 可通过对应云产品的查询实例接口获取实例 ID。目前支持实例类型： - CVM - Lighthouse - TAT 托管实例 示例值：["ins-zcxx4ga1"]
CommandName	否	String	命令名称。名称仅支持中文、英文、数字、下划线、分隔符"-"、小数点，最大长度不能超60个字节。 示例值：tat-command
Description	否	String	命令描述。不超过120字符。 示例值：this is a command
CommandType	否	String	命令类型，目前支持取值：SHELL、POWERSHELL、BAT。默认：SHELL。 示例值：SHELL
WorkingDirectory	否	String	命令执行路径，对于 SHELL 命令默认为 /root，对于 POWERSHELL 命令默认为 C:\Program Files\qcloud\tat_agent\workdir。 示例值：/root
Timeout	否	Integer	命令超时时间，默认60秒。取值范围[1, 86400]。 示例值：60
SaveCommand	否	Boolean	是否保存命令，取值范围： • true：保存 • false：不保存

参数名称	必选	类型	描述
			默认为 false。 示例值：false
EnableParameter	否	Boolean	是否启用自定义参数功能。 一旦创建，此值不提供修改。 取值范围： • true：启用 • false：不启用 默认值：false。 示例值：false
DefaultParameters	否	String	启用自定义参数功能时，自定义参数的默认取值。字段类型为json encoded string。 如：{"varA": "222"}。 key为自定义参数名称，value为该参数的默认取值。kv均为字符串型。 仅在命令的 EnableParameter 为 true 时，才允许设置此参数。 参数不支持同时指定 `DefaultParameters` 和 `DefaultParameterConfs` 。 如果 Parameters 未提供，将使用这里的默认值进行替换。 自定义参数最多20个。 自定义参数名称需符合以下规范：字符数目上限64，可选范围【a-zA-Z0-9-_】。 示例值：{"varA": "222"}
DefaultParameterConfs.N	否	Array of DefaultParameterConf	自定义参数数组。如果 Parameters 未提供，将使用这里的默认值进行替换。 自定义参数最多20个。 如果 Parameters 未提供，将使用这里的默认值进行替换。 仅在命令的 EnableParameter 为 true 时，才允许设置此参数。 参数不支持同时指定 `DefaultParameters` 和 `DefaultParameterConfs` 。 示例值：[{ "ParameterName": "test01", "ParameterValue": "12345", "ParameterDescription": "for test01" }, { "ParameterName": "test02", "ParameterValue": "12345", "ParameterDescription": "for test02"}]
Parameters	否	String	Command 的自定义参数。字段类型为json encoded string。如：{"varA": "222"}。 key为自定义参数名称，value为该参数的默认取值。kv均为字符串型。 仅在命令的 EnableParameter 为 true 时，才允许设置此参数。 如果未提供该参数取值，将使用 DefaultParameters 或 DefaultParameterConfs 进行替换。 自定义参数最多20个。 自定义参数名称需符合以下规范：字符数目上限64，可选范围【a-zA-Z0-9-_】。 示例值：{"varA": "222"}
Tags.N	否	Array of Tag	如果保存命令，可为命令设置标签。列表长度不超过10。 示例值：[{ "Key": "test", "Value": "test" }]
Username	否	String	在 CVM 或 Lighthouse 实例中执行命令的用户名称。 使用最小权限执行命令是权限管理的最佳实践，建议您以普通用户身份执行云助手命令。 默认情况下，在 Linux 实例中以 root 用户执行命令；在Windows 实例中以 System 用户执行命令。 示例值：root
OutputCOSBucketUrl	否	String	指定日志上传的cos bucket 地址，必须以https开头，如 https://BucketName-123454321.cos.ap-beijing.myqcloud.com。 示例值：https://BucketName-123454321.cos.ap-beijing.myqcloud.com
OutputCOSKeyPrefix	否	String	指定日志在cos bucket中的目录，目录命名有如下规则： 1. 可用数字、中英文和可见字符的组合，长度最多为60。 2. 用 / 分割路径，可快速创建子目录。 3. 不允许连续 / ；不允许以 / 开头；不允许以..作为文件夹名称。 示例值：aa/bb/cc

3. 输出参数

参数名称	类型	描述
CommandId	String	命令ID。

参数名称	类型	描述
		示例值：cmd-5oa8jajm
InvocationId	String	执行活动ID。 示例值：inv-mp6m9vmu
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 在 CVM 批量执行命令

输入示例

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

{
"CommandName": "run-command",
"SaveCommand": false,
"Description": "whoami",
"Content": "d2hvYW1p",
"CommandType": "SHELL",
"WorkingDirectory": "/root/",
"Timeout": 60,
"InstanceIds": [
"ins-zj0f57ew",
"ins-zj0f57ex",
"ins-zj0f57ev"
]
}
```

输出示例

```
{
"Response": {
"RequestId": "f3e3a951-56b1-4042-af23-ba50223a8f60",
"CommandId": "cmd-5oa8jajm",
"InvocationId": "inv-mp6m9vmu"
}
}
```

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.CVMError	调用 CVM 失败。
FailedOperation.LighthouseError	调用 Lighthouse 失败。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameter.InvalidUsername	无效用户名。
InvalidParameter.ParameterNameDuplicated	参数名称重复。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.AgentUnsupportedCommandType	Agent不支持此命令类型。
InvalidParameterValue.CommandContentInvalid	Command 内容无效。
InvalidParameterValue.CommandNameDuplicated	Command 名称重复。
InvalidParameterValue.InconsistentInstance	实例类型不一致。
InvalidParameterValue.InvalidCommandName	Command 名称无效。
InvalidParameterValue.InvalidContent	命令内容无效。
InvalidParameterValue.InvalidInstanceId	实例ID无效。
InvalidParameterValue.InvalidOutputCOSBucketUrl	OutputCOSBucketUrl 无效。
InvalidParameterValue.InvalidOutputCOSKeyPrefix	OutputCOSKeyPrefix 无效。
InvalidParameterValue.InvalidUsername	用户名不合法。
InvalidParameterValue.InvalidWorkingDirectory	命令执行路径不合法。
InvalidParameterValue.LackOfParameterInfo	已启用自定义参数功能，但缺失自定义参数信息。
InvalidParameterValue.LimitExceeded	超过参数限制。
InvalidParameterValue.ParameterDisabled	未启用自定义参数功能。
InvalidParameterValue.ParameterInvalidJsonFormat	参数为非法 json string 格式。
InvalidParameterValue.ParameterKeyContainsInvalidChar	参数 Key 包含非法字符。
InvalidParameterValue.ParameterKeyDuplicated	参数 Key 重复。
InvalidParameterValue.ParameterKeyLenExceeded	参数 Key 过长。
InvalidParameterValue.ParameterNumberExceeded	参数数目过多。

错误码	描述
InvalidParameterValue.ParameterValueNotString	参数 Value 非 string 类型。
InvalidParameterValue.Range	参数取值范围不合法。
InvalidParameterValue.SupportParametersOnlyIfEnableParameter	未启用自定义参数功能。
InvalidParameterValue.TooLong	长度超过限制。
MissingParameter	缺少参数错误。
OperationDenied	操作被拒绝。
ResourceNotFound.InstanceNotFound	实例不存在。
ResourceNotFound.RoleNotFound	角色不存在。
ResourceUnavailable	资源不可用。
ResourceUnavailable.AgentNotInstalled	Agent 未安装。
ResourceUnavailable.AgentStatusNotOnline	Agent 不在线。
ResourceUnavailable.InstanceStateNotRunning	实例未处于运行中。
ResourceUnavailable.LighthouseUnsupportedRegion	Lighthouse 尚不支持指定的地域。
ResourceUnavailable.UserHasNoQuotaCode	用户配额已用完
UnauthorizedOperation.AssumeRoleUnauthorized	角色扮演未授权。
UnauthorizedOperation.CamAuthFailed	CAM鉴权失败。
UnauthorizedOperation.InvalidToken	Token 无效。
UnauthorizedOperation.MFAExpired	Multi-Factor Authentication(MFA) 过期。
UnauthorizedOperation.MFANotFound	Multi-Factor Authentication(MFA) 不存在。

取消命令执行

最近更新时间：2025-04-25 01:54:41

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

取消一台或多台实例执行的命令

- 如果命令还未下发到agent，任务状态处于PENDING、DELIVERING、DELIVER_DELAYED，取消后任务状态是CANCELLED
- 如果命令已下发到agent，任务状态处于RUNNING， 取消后任务状态是TERMINATED

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：CancelInvocation。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
InvocationId	是	String	执行活动ID。 可通过 DescribeInvocations(查询执行活动) 接口获取。 示例值：inv-xxxxxxxx
InstanceIds.N	否	Array of String	实例ID列表，上限100。 可通过对应云产品的查询实例接口获取实例 ID。目前支持实例类型： - CVM - Lighthouse - TAT 托管实例 示例值：["ins-xxxxxxxx"]

3. 输出参数

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

4. 示例

示例1 取消命令执行

输入示例

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

```
{
"InvocationId": "inv-ib2l1d37",
"InstanceIds": [
"ins-zcewfho0"
]
}
```

输出示例

```
{
"Response": {
"RequestId": "b27421da-0ad4-4e11-b694-cda143606701"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.InstanceIsNotRelatedToInvocation	实例ID与执行活动无关。
InvalidParameterValue.InvalidInstanceId	实例ID无效。
InvalidParameterValue.InvalidInvocationId	不合法的执行活动ID。
InvalidParameterValue.LimitExceeded	超过参数限制。
InvalidParameterValue.TooLong	长度超过限制。

错误码	描述
LimitExceeded	超过配额限制。
MissingParameter	缺少参数错误。
ResourceNotFound.InvocationNotFound	执行活动未找到。
UnauthorizedOperation.CamAuthFailed	CAM鉴权失败。
UnauthorizedOperation.InvalidToken	Token 无效。
UnauthorizedOperation.MFAExpired	Multi-Factor Authentication(MFA) 过期。
UnauthorizedOperation.MFANotFound	Multi-Factor Authentication(MFA) 不存在。
UnknownParameter	未知参数错误。

查询执行活动

最近更新时间：2025-04-25 01:54:40

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于查询执行活动详情。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeInvocations。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
InvocationIds.N	否	Array of String	执行活动ID列表，每次请求的上限为100。参数不支持同时指定 InvocationIds 和 Filters。 示例值：["inv-xxxxxxxx"]
Filters.N	否	Array of Filter	过滤条件。 • invocation-id - String - 是否必填：否 -（过滤条件）按照执行活动ID过滤。 • command-id - String - 是否必填：否 -（过滤条件）按照命令ID过滤。 • command-created-by - String - 是否必填：否 -（过滤条件）按照执行的命令类型过滤，取值为 TAT 或 USER，TAT 代表公共命令，USER 代表由用户创建的命令。 • instance-kind - String - 是否必填：否 -（过滤条件）按照运行实例类型过滤，取值为 CVM 或 LIGHTHOUSE，CVM 代表实例为云服务器， LIGHTHOUSE 代表实例为轻量应用服务器。 每次请求的 Filters 的上限为10， Filter.Values 的上限为5。参数不支持同时指定 InvocationIds 和 Filters 。 示例值：[{ "Name": "invocation-id", "Values": ["inv-zcewfho0"] }]
Limit	否	Integer	返回数量，默认为20，最大值为100。关于 Limit 的更进一步介绍请参考 API 简介中的相关小节。 示例值：10
Offset	否	Integer	偏移量，默认为0。关于 Offset 的更进一步介绍请参考 API 简介中的相关小节。 示例值：0

3. 输出参数

参数名称	类型	描述
TotalCount	Integer	符合条件的执行活动总数。 示例值：1
InvocationSet	Array of Invocation	执行活动列表。

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 查询执行活动

输入示例

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

{
"Offset": 0,
"Limit": 1,
"InvocationIds": [
"inv-q4zp60g0"
]
}
```

输出示例

```
{
"Response": {
"RequestId": "958a3603-d0c3-4c97-a37d-6bc24eacddac",
"TotalCount": 1,
"InvocationSet": [
{
"CommandId": "cmd-9dxzty3m",
"CommandContent": "cHdk",
"CommandType": "SHELL",
"Timeout": 60,
"InvocationSource": "USER",
"WorkingDirectory": "/root",
"InvocationId": "inv-q4zp60g0",
"InstanceKind": "CVM",
"InvocationStatus": "SUCCESS",
"Description": "Test Invocation",
"Parameters": "",
"DefaultParameters": "{\"k1\":\"v1\"}",
"Username": "root",
"OutputCOSKeyPrefix": "cosprefix",
"OutputCOSBucketUrl": "https://example-123456789.cos.ap-beijing.myqcloud.com",
"InvocationTaskBasicInfoSet": [
{
"InvocationTaskId": "invt-kakne8h2",
"TaskStatus": "SUCCESS",
"InstanceId": "ins-zj0f57ew"
},
{
"InvocationTaskId": "invt-jc2onrxm",
"TaskStatus": "SUCCESS",
```

```
"InstanceId": "ins-zj0f57ex"
},
{
"InvocationTaskId": "invt-6xmuisyq",
"TaskStatus": "SUCCESS",
"InstanceId": "ins-zj0f57ev"
}
],
"StartTime": "2020-11-09T07:21:59+00:00",
"EndTime": "2020-11-09T07:22:08+00:00",
"CreatedTime": "2020-11-09T07:21:59+00:00",
"UpdatedTime": "2020-11-09T07:22:08+00:00"
}
]
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameter.ConflictParameter	参数冲突。
InvalidParameterValue.InvalidCommandId	CommandId 无效。
InvalidParameterValue.InvalidFilter	Filter 无效。
InvalidParameterValue.InvalidInvocationId	不合法的执行活动ID。
InvalidParameterValue.LimitExceeded	超过参数限制。
LimitExceeded.FilterValueExceeded	填写的 Filter 取值过多。

错误码	描述
UnauthorizedOperation.CamAuthFailed	CAM鉴权失败。
UnauthorizedOperation.InvalidToken	Token 无效。
UnauthorizedOperation.MFAExpired	Multi-Factor Authentication(MFA) 过期。
UnauthorizedOperation.MFANotFound	Multi-Factor Authentication(MFA) 不存在。

查询执行任务

最近更新时间：2025-04-25 01:54:40

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于查询执行任务详情。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeInvocationTasks。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
InvocationTaskIds.N	否	Array of String	执行任务ID列表，每次请求的上限为100。参数不支持同时指定 InvocationTaskIds 和 Filters。 示例值：["invt-xxxxxxxx"]
Filters.N	否	Array of Filter	过滤条件。 - invocation-task-id - String - 是否必填：否 -（过滤条件）按照执行任务ID过滤。 - invocation-id - String - 是否必填：否 -（过滤条件）按照执行活动ID过滤。可通过 DescribeInvocations(查询执行活动) 接口获取。 - instance-id - String - 是否必填：否 -（过滤条件）按照实例ID过滤。可通过对应云产品的查询实例接口获取实例 ID。目前支持实例类型： CVM、Lighthouse、TAT 托管实例 - command-id - String - 是否必填：否 -（过滤条件）按照命令ID过滤。可通过 DescribeCommands(查询命令详情) 接口获取。 每次请求的 Filters 的上限为10， Filter.Values 的上限为5。参数不支持同时指定 InvocationTaskIds 和 Filters 。 示例值：[{"Name": "instance-id", "Values": ["ins-zcewfho0"]}]
Limit	否	Integer	返回数量，默认为20，最大值为100。关于 Limit 的更进一步介绍请参考 API 简介中的相关小节。 示例值：10
Offset	否	Integer	偏移量，默认为0。关于 Offset 的更进一步介绍请参考 API 简介中的相关小节。 示例值：0
HideOutput	否	Boolean	是否隐藏命令输出结果，取值范围： - true：隐藏输出 - false：不隐藏 默认为 true。 示例值：true

3. 输出参数

参数名称	类型	描述
TotalCount	Integer	符合条件的执行任务总数。 示例值：3
InvocationTaskSet	Array of InvocationTask	执行任务列表。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 根据执行活动ID查询所有执行任务详情

输入示例

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

{
"Offset": 0,
"Limit": 10,
"HideOutput": false,
"Filters": [
{
"Name": "invocation-id",
"Values": [
"inv-1vll7hda"
]
}
]
}
```

输出示例

```
{
"Response": {
"RequestId": "a1df9725-51c6-466d-a038-4c86461a8e26",
"TotalCount": 1,
"InvocationTaskSet": [
{
"CommandId": "cmd-13axrtuq",
"CommandDocument": {
"Content": "d2hvYW1p",
"CommandType": "SHELL",
"Timeout": 1,
"Username": "root",
"WorkingDirectory": "/root/",
"OutputCOSBucketUrl": "https://example-123456789.cos.ap-beijing.myqcloud.com",
"OutputCOSKeyPrefix": "datalog"
},
"InvocationId": "inv-1vll7hda",
"InvocationTaskId": "invt-08oe5fe2",
"TaskStatus": "SUCCESS",
```

```
"InstanceId": "ins-zj0f57ex",
"TaskResult": {
"ExitCode": 0,
"Output": "cm9vdAo=",
"Dropped": 0,
"OutputUploadCOSErrorInfo": "",
"OutputUrl": "",
"ExecStartTime": "2020-11-05T07:49:58+00:00",
"ExecEndTime": "2020-11-05T07:50:04+00:00"
},
"ErrorInfo": "",
"InvocationSource": "user",
"StartTime": "2020-11-05T07:49:58+00:00",
"EndTime": "2020-11-05T07:50:04+00:00",
"CreatedTime": "2020-11-05T07:49:56+00:00",
"UpdatedTime": "2020-11-05T07:50:06+00:00"
}
]
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameter.ConflictParameter	参数冲突。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.InvalidCommandId	CommandId 无效。

错误码	描述
InvalidParameterValue.InvalidFilter	Filter 无效。
InvalidParameterValue.InvalidInstanceId	实例ID无效。
InvalidParameterValue.InvalidInvocationId	不合法的执行活动ID。
InvalidParameterValue.InvalidInvocationTaskId	不合法的执行任务ID。
UnauthorizedOperation.CamAuthFailed	CAM鉴权失败。
UnauthorizedOperation.InvalidToken	Token 无效。
UnauthorizedOperation.MFAExpired	Multi-Factor Authentication(MFA) 过期。
UnauthorizedOperation.MFANotFound	Multi-Factor Authentication(MFA) 不存在。
UnsupportedOperation	操作不支持。

查询客户端状态

最近更新时间：2025-04-28 01:58:54

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于查询自动化助手客户端的状态。

默认接口请求频率限制：60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：DescribeAutomationAgentStatus。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
InstanceIds.N	否	Array of String	待查询的实例ID列表。 可通过对应云产品的查询实例接口获取实例 ID。目前支持实例类型：CVM、Lighthouse、TAT 托管实例。 每次请求的上限为 100。 参数不支持同时指定 InstanceIds 和 Filters 。 示例值：["lhins-ar5wyn4x"]
Filters.N	否	Array of Filter	- agent-status - String - 是否必填：否 -（过滤条件）按照agent状态过滤，取值：Online 在线，Offline 离线。 - environment - String - 是否必填：否 -（过滤条件）按照agent运行环境查询，取值：Linux, Windows。 - instance-id - String - 是否必填：否 -（过滤条件）按照实例ID过滤。 可通过对应云产品的查询实例接口获取实例 ID。目前支持实例类型：CVM、Lighthouse、TAT 托管实例。 每次请求的 Filters 的上限为10， Filter.Values 的上限为5。参数不支持同时指定 InstanceIds 和 Filters 。 示例值：[{ "Name": "instance-id", "Values": ["ins-zcewfho0"] }]
Limit	否	Integer	返回数量，默认为20，最大值为100。关于 Limit 的更进一步介绍请参考 API 简介中的相关小节。 示例值：10
Offset	否	Integer	偏移量，默认为0。关于 Offset 的更进一步介绍请参考 API 简介中的相关小节。 示例值：0

3. 输出参数

参数名称	类型	描述
AutomationAgentSet	Array of AutomationAgentInfo	Agent 信息列表。
TotalCount	Integer	符合条件的 Agent 总数。 示例值：1
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询客户端状态

输入示例

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

{
"InstanceIds": [
"lhins-ar5wyn4x"
]
}
```

输出示例

```
{
"Response": {
"RequestId": "520e7b93-904d-44de-a2dd-8e4c545be4ce",
"AutomationAgentSet": [
{
"InstanceId": "lhins-ar5wyn4x",
"Version": "0.1.0",
"LastHeartbeatTime": "2020-11-16T12:05:44+00:00",
"AgentStatus": "Online",
"Environment": "Linux",
"SupportFeatures": [
"SESSION_MANAGER"
]
}
],
"TotalCount": 1
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameter.ConflictParameter	参数冲突。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.InvalidFilter	Filter 无效。
InvalidParameterValue.InvalidInstanceId	实例ID无效。
InvalidParameterValue.LimitExceeded	超过参数限制。
UnauthorizedOperation.CamAuthFailed	CAM鉴权失败。
UnauthorizedOperation.InvalidToken	Token 无效。
UnauthorizedOperation.MFAExpired	Multi-Factor Authentication(MFA) 过期。
UnauthorizedOperation.MFANotFound	Multi-Factor Authentication(MFA) 不存在。
UnsupportedOperation	操作不支持。

定时执行相关接口
创建执行器

最近更新时间：2025-04-28 01:58:56

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于创建执行器。

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：CreateInvoker。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
Name	是	String	执行器名称。长度不超过 120 字符。 示例值：invoker-test
Type	是	String	执行器类型。 可选取值（当前仅支持一种）： - SCHEDULE：周期类型执行器。 示例值：SCHEDULE
CommandId	是	String	远程命令ID。 可通过 DescribeCommands(查询命令详情) 接口获取。 示例值：cmd-k4ypdij0
InstanceIds.N	是	Array of String	触发器关联的实例ID。列表上限 100。 可通过对应云产品的查询实例接口获取实例 ID。目前支持实例类型：CVM、Lighthouse、TAT 托管实例。 实例需要安装 TAT 客户端，且客户端为 Online 状态。可通过 DescribeAutomationAgentStatus(查询客户端状态) 接口查询客户端状态。 示例值：["ins-yx05ky8j"]
Username	否	String	命令执行用户。长度不超过 256 字符。 示例值：root
Parameters	否	String	命令自定义参数。字段类型为 JSON encode string。 仅在 CommandId 所指命令的 EnableParameter 为 true 时，才允许设置此参数。可通过 DescribeCommands(查询命令详情) 接口获取命令的 EnableParameter 设置。 示例值：{"var": "A"}
ScheduleSettings	否	ScheduleSettings	周期执行器设置。 当执行器类型为 SCHEDULE 时，必须指定此参数。 示例值：{"Policy":"ONCE","InvokeTime":"2021-09-01T00:00:00+08:00"}

3. 输出参数

参数名称	类型	描述
InvokerId	String	执行器ID。 示例值：ivk-b7s3qa5l
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 创建单次执行器

北京时间 2021-09-01 00:00:00 在实例 ins-yx05ky8j 执行命令 cmd-m7uma92n

输入示例

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

{
"Name": "test-invoker",
"CommandId": "cmd-m7uma92n",
"InstanceIds": [
"ins-yx05ky8j"
],
"Type": "SCHEDULE",
"ScheduleSettings": {
"Policy": "ONCE",
"InvokeTime": "2021-09-01T00:00:00+08:00"
}
}
```

输出示例

```
{
"Response": {
"RequestId": "97268137-e0f7-477d-833b-766273d0ea47",
"InvokerId": "ivk-le1g3x2h"
}
}
```

示例2 创建周期执行命令

在北京时间每月1日零点，在实例 ins-yx05ky8j 执行命令 cmd-m7uma92n

输入示例

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

{
```

```
"Name": "cron-invoker",
"CommandId": "cmd-m7uma92n",
"InstanceIds": [
"ins-yx05ky8j"
],
"Type": "SCHEDULE",
"ScheduleSettings": {
"Policy": "RECURRENCE",
"Recurrence": "0 0 1 * *"
}
}
```

输出示例

```
{
"Response": {
"RequestId": "d1d7ce29-b6ac-436d-93e0-b454f096cc50",
"InvokerId": "ivk-n0t6rxtv"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue.ID	ID 格式错误。
InvalidParameterValue.InconsistentID	ID 数组中，ID 格式错误或格式不一致。

错误码	描述
InvalidParameterValue.InconsistentInstance	实例类型不一致。
InvalidParameterValue.InvalidCommandId	CommandId 无效。
InvalidParameterValue.InvalidCronExpression	Crontab 表达式无效。
InvalidParameterValue.InvalidInstanceId	实例ID无效。
InvalidParameterValue.InvalidTimeFormat	无效的时间格式。
InvalidParameterValue.InvokeTimeExpired	调用时间已过期。
InvalidParameterValue.ParameterInvalidJsonFormat	参数为非法 json string 格式。
InvalidParameterValue.Range	参数取值范围不合法。
InvalidParameterValue.TooLong	长度超过限制。
ResourceNotFound.CommandNotFound	命令不存在。
ResourceNotFound.InstanceNotFound	实例不存在。
ResourceUnavailable.AgentNotInstalled	Agent 未安装。
ResourceUnavailable.AgentStatusNotOnline	Agent 不在线。

修改执行器

最近更新时间：2025-04-28 01:58:55

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于修改执行器。

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：ModifyInvoker。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
InvokerId	是	String	待修改的执行器ID。 可通过 DescribeInvokers(查询执行器) 接口获取。 示例值：ivk-n0t6rxtv
Name	否	String	待修改的执行器名称。长度不超过 120 字符。 示例值：invoker-test
Type	否	String	待修改的执行器类型。 可选取值（当前仅支持一种）： - SCHEDULE：周期类型执行器。 示例值：SCHEDULE
CommandId	否	String	待修改的命令ID。 可通过 DescribeCommands(查询命令详情) 接口获取。 示例值：cmd-m7uma92n
Username	否	String	待修改的用户名。长度不超过 256 字符。 示例值：root
Parameters	否	String	待修改的自定义参数。字段类型为 JSON encode string。 仅在 CommandId 所指命令的 EnableParameter 为 true 时，才允许设置此参数。可通过 DescribeCommands(查询命令详情) 接口获取命令的 EnableParameter 设置。 示例值：{"var": "A"}
InstanceIds.N	否	Array of String	待修改的实例ID列表。列表长度上限100。 可通过对应云产品的查询实例接口获取实例 ID。目前支持实例类型：CVM、Lighthouse、TAT 托管实例。 实例需要安装 TAT 客户端，且客户端为 Online 状态。可通过 DescribeAutomationAgentStatus(查询客户端状态) 接口查询客户端状态。 示例值：["ins-yx05ky8j"]
ScheduleSettings	否	ScheduleSettings	待修改的周期执行器设置。

参数名称	必选	类型	描述
			要将执行器类型修改为 SCHEDULE 时，必须指定此参数。 示例值：{ "Policy":"ONCE", "InvokeTime":"2021-09-01T00:00:00+08:00" }

3. 输出参数

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

4. 示例

示例1 修改执行器

输入示例

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

{
"InvokerId": "ivk-b7s3qa5l",
"Parameters": "{\"var\": \"1\"}"
}
```

输出示例

```
{
"Response": {
"RequestId": "7f3acde8-196d-4be4-9fa7-359f79b026bc"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
InternalError	内部错误。
InvalidParameterValue.ID	ID 格式错误。
InvalidParameterValue.InconsistentID	ID 数组中，ID 格式错误或格式不一致。
InvalidParameterValue.InconsistentInstance	实例类型不一致。
InvalidParameterValue.InvalidCronExpression	Crontab 表达式无效。
InvalidParameterValue.InvalidInstanceId	实例ID无效。
InvalidParameterValue.InvalidInvokerId	InvokerId 无效。
InvalidParameterValue.InvalidTimeFormat	无效的时间格式。
InvalidParameterValue.InvokeTimeExpired	调用时间已过期。
InvalidParameterValue.LimitExceeded	超过参数限制。
InvalidParameterValue.ParameterDisabled	未启用自定义参数功能。
InvalidParameterValue.ParameterInvalidJsonFormat	参数为非法 json string 格式。
InvalidParameterValue.Range	参数取值范围不合法。
InvalidParameterValue.TooLong	长度超过限制。
ResourceNotFound.CommandNotFound	命令不存在。
ResourceNotFound.InstanceNotFound	实例不存在。
ResourceUnavailable.AgentNotInstalled	Agent 未安装。

删除执行器

最近更新时间：2025-04-28 01:58:56

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于删除执行器。

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：DeleteInvoker。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
InvokerId	是	String	待删除的执行器ID。 可通过 DescribeInvokers(查询执行器) 接口获取。 示例值：ivk-b7s3qa5l

3. 输出参数

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

4. 示例

示例1 删除执行器

输入示例

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

{
"InvokerId": "ivk-b7s3qa5l"
}
```

输出示例

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

```
"RequestId": "bc16e010-7b50-4661-88e8-4d2c77d15558"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
InvalidParameterValue.ID	ID 格式错误。
InvalidParameterValue.InvalidInvokerId	InvokerId 无效。
ResourceNotFound	资源不存在。

启用执行器

最近更新时间：2025-04-28 01:58:55

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于启用执行器。

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：EnableInvoker。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
InvokerId	是	String	待启用的执行器ID。 可通过 DescribeInvokers(查询执行器) 接口获取。 示例值：ivk-b7s3qa5l

3. 输出参数

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

4. 示例

示例1 启用执行器

输入示例

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

{
"InvokerId": "ivk-b7s3qa5l"
}
```

输出示例

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

```
"RequestId": "64119f65-18f5-47f3-a8d2-60d9fee593bf"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
InvalidParameterValue.ID	ID 格式错误。
InvalidParameterValue.InvalidInvokerId	InvokerId 无效。
ResourceNotFound	资源不存在。

停用执行器

最近更新时间：2025-04-28 01:58:55

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于停止执行器。

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：DisableInvoker。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
InvokerId	是	String	待停止的执行器ID。 可通过 DescribeInvokers(查询执行器) 接口获取。 示例值：ivk-b7s3qa5l

3. 输出参数

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

4. 示例

示例1 停止执行器

输入示例

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

{
"InvokerId": "ivk-b7s3qa5l"
}
```

输出示例

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

```
"RequestId": "a6c5b0bd-4273-47c0-8d34-c32822f3ccb0"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
InvalidParameterValue.ID	ID 格式错误。
InvalidParameterValue.InvalidInvokerId	InvokerId 无效。
ResourceNotFound	资源不存在。

查询执行器

最近更新时间：2025-04-28 01:58:56

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于查询执行器信息。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeInvokers。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
InvokerIds.N	否	Array of String	执行器 ID 列表。 每次请求的上限为 100。 参数不支持同时指定 `InvokerIds` 和 `Filters` 。 示例值：["ivk-b7s3qa5l"]
Filters.N	否	Array of Filter	过滤条件： - invoker-id - String - 是否必填：否 -（过滤条件）按执行器ID过滤。 - command-id - String - 是否必填：否 -（过滤条件）按命令ID过滤。可通过 DescribeCommands(查询命令详情) 接口获取。 - invoker-type - String - 是否必填：否 -（过滤条件）按执行器类型过滤。目前仅支持 SCHEDULE 一种。 每次请求的 `Filters` 的上限为 10，`Filter.Values` 的上限为 5。参数不支持同时指定 `InvokerIds` 和 `Filters` 。 示例值：[{"Name": "command-id", "Values": ["cmd-nrncl292"]}]
Limit	否	Integer	返回数量，默认为20，最大值为100。 示例值：10
Offset	否	Integer	偏移量，默认为0。 示例值：0

3. 输出参数

参数名称	类型	描述
TotalCount	Integer	满足条件的执行器数量。 示例值：10
InvokerSet	Array of Invoker	执行器信息。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询执行器

输入示例

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

{
"Filters": [
{
"Name": "invoker-id",
"Values": [
"ivk-b7s3qa5l"
]
}
]
}
```

输出示例

```
{
"Response": {
"RequestId": "a4c828cf-31c9-42d2-a558-2ad340a76ef0",
"TotalCount": 1,
"InvokerSet": [
{
"InvokerId": "ivk-b7s3qa5l",
"Name": "test-invoker",
"Type": "SCHEDULE",
"CommandId": "cmd-m7uma92n",
"Username": "root",
"Parameters": "{\"var\": \"1\"}",
"InstanceIds": [
"ins-yx05ky8j"
],
"Enable": false,
"ScheduleSettings": {
"Policy": "SCHEDULE",
"Recurrence": "0 0 1 * *",
"InvokeTime": "2021-08-30T06:42:02Z"
},
"CreatedTime": "2021-08-30T06:42:02Z",
"UpdatedTime": "2021-09-09T12:07:00Z"
}
]
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
InvalidParameter.ConflictParameter	参数冲突。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.InconsistentID	ID 数组中，ID 格式错误或格式不一致。
InvalidParameterValue.InvalidFilter	Filter 无效。
InvalidParameterValue.InvalidInvokerId	InvokerId 无效。
InvalidParameterValue.LimitExceeded	超过参数限制。
InvalidParameterValue.TooLarge	参数取值过大。
InvalidParameterValue.TooSmall	参数取值过小。

查询执行器执行记录

最近更新时间：2025-04-28 01:58:56

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于查询执行器的执行记录。

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：DescribeInvokerRecords。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
InvokerIds.N	否	Array of String	执行器ID列表。列表上限 100。 可通过 DescribeInvokers(查询执行器) 接口获取。 示例值：["ivk-b7s3qa5l"]
Limit	否	Integer	返回数量，默认为20，最大值为100。 示例值：10
Offset	否	Integer	偏移量，默认为0。 示例值：0

3. 输出参数

参数名称	类型	描述
TotalCount	Integer	符合条件的历史记录数量。 示例值：10
InvokerRecordSet	Array of InvokerRecord	执行器执行历史记录。 示例值：[{"InvocationId": "inv-sca38601ud", "InvokeTime": "2024-10-25T22:24:13Z", "InvokerId": "ivk-ngqk4bda", "Reason": "start an invocation at scheduled time.", "Result": "SUCCESS" }]
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询执行器执行记录

输入示例

```
POST / HTTP/1.1
Host: tat.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DescribeInvokerRecords
```

```
<公共请求参数>

{
"InvokerIds": [
"ivk-b7s3qa5l"
],
"Offset": 0,
"Limit": 2
}
```

输出示例

```
{
"Response": {
"RequestId": "0398dcea-b3de-4ec9-8e78-976191e0a94f",
"TotalCount": 2,
"InvokerRecordSet": [
{
"InvokerId": "ivk-b7s3qa5l",
"Reason": "start an invocation at scheduled time.",
"InvocationId": "inv-l9l4orf1",
"Result": "SUCCESS",
"InvokeTime": "2021-09-06T09:33:05Z"
},
{
"InvokerId": "ivk-b7s3qa5l",
"Reason": "start an invocation at scheduled time.",
"InvocationId": "inv-7ojgezbd",
"Result": "SUCCESS",
"InvokeTime": "2021-09-06T09:30:05Z"
}
]
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.InconsistentID	ID 数组中，ID 格式错误或格式不一致。
InvalidParameterValue.InvalidInvokerId	InvokerId 无效。
InvalidParameterValue.TooLarge	参数取值过大。
InvalidParameterValue.TooLong	长度超过限制。
InvalidParameterValue.TooSmall	参数取值过小。
UnsupportedOperation	操作不支持。

托管实例相关接口
创建注册码

最近更新时间：2025-04-25 01:54:42

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

接口用于创建注册码。

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：CreateRegisterCode。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
Description	否	String	注册码描述。最大长度为 128 字符。 示例值：此注册码用于注册托管实例。
InstanceNamePrefix	否	String	注册实例名称前缀。最大长度为 32 字符。 示例值：register-test
RegisterLimit	否	Integer	该注册码允许注册的实例数目。默认值为 10，最小值为 1，最大值为 10000。 示例值：10
EffectiveTime	否	Integer	该注册码的有效时间，单位为小时。默认值为 4。 - 若传入值小于等于 99999，则以小时为单位设置有效时间。 - 若传入值大于 99999，则设置为长期有效。 示例值：4
IpAddressRange	否	String	限制注册码只能从 IpAddressRange 所描述公网出口进行注册。 默认为空，即无任何限制。 取值应为标准 IPv4 或 CIDRv4 格式。例如 192.168.1.1 或 192.168.0.0/16。 示例值：133.12.234.0/24

3. 输出参数

参数名称	类型	描述
RegisterCodeId	String	注册码ID。 示例值：d0b7xxxx-a6xx-40x9-898x-44c9f508axxx
RegisterCodeValue	String	注册码值。 示例值：5xx127965b0b498195xxxxx80ad926c53e1xxxx3d85e41e0ad93555941111111
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 创建注册码-带参数

创建注册码-带参数。

输入示例

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

{
"Description": "webserver",
"InstanceNamePrefix": "webserver",
"RegisterLimit": 10,
"EffectiveTime": 4,
"IpAddressRange": "192.168.10.1/10"
}
```

输出示例

```
{
"Response": {
"RegisterCodeId": "9b5rb72b-418f-4a81-b32e-f8bc80eef678",
"RegisterCodeValue": "21e9b871e123427dbee18b0cf3e0d766f91d81a9902843b7b046d7b9ffb9d45a",
"RequestId": "3f5bf8de-a51a-4ac0-8d95-ad56702cab1f"
}
}
```

示例2 创建注册码-无参数

创建注册码-无参数。

输入示例

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

{}
```

输出示例

```
{
"Response": {
"RegisterCodeId": "9b5rb72b-418f-4a81-b32e-f8bc80eef678",
"RegisterCodeValue": "21e9b871e123427dbee18b0cf3e0d766f91d81a9902843b7b046d7b9ffb9d45a",
"RequestId": "2e8bf8de-a51a-4ac0-8d95-ad56702cab1f"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter	参数错误。
InvalidParameterValue.TooLarge	参数取值过大。
InvalidParameterValue.TooLong	长度超过限制。
InvalidParameterValue.TooSmall	参数取值过小。
ResourceNotFound.RoleNotFound	角色不存在。
UnauthorizedOperation.AssumeRoleUnauthorized	角色扮演未授权。

批量禁用注册码

最近更新时间：2025-04-25 01:54:41

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于批量禁用注册码。

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：DisableRegisterCodes。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
RegisterCodeIds.N	是	Array of String	注册码ID。 可通过 DescribeRegisterCodes(查询注册码) 接口获取。 示例值：["d0b7xxxx-a6xx-40x9-898x-44c9f508axxx"]

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 禁用注册码

根据注册码ID禁用注册码。

输入示例

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

{
"RegisterCodeIds": [
"8cca2d3b-7ac3-422a-98f0-8a5bc17cdc38"
]
}
```

输出示例

```
{
"Response": {
"RequestId": "e0f011ac-6949-4726-a7d6-b28540f9d729"
}
}
```

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
InvalidParameterValue.InvalidRegisterCodeId	无效的注册码ID。
InvalidParameterValue.ParameterKeyDuplicated	参数 Key 重复。
MissingParameter	缺少参数错误。
ResourceNotFound.RegisterCodesNotFoundCode	查询不到注册码。
ResourceNotFound.RoleNotFound	角色不存在。
UnauthorizedOperation.AssumeRoleUnauthorized	角色扮演未授权。

批量删除注册码

最近更新时间：2025-04-25 01:54:42

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于批量删除注册码。

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：DeleteRegisterCodes。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
RegisterCodeIds.N	是	Array of String	注册码ID列表。限制输入的注册码ID数量大于0小于100。 可通过 DescribeRegisterCodes(查询注册码) 接口获取。 示例值：["d0b7xxxx-a6xx-40x9-898x-44c9f508axxx"]

3. 输出参数

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

4. 示例

示例1 删除注册码

根据注册码ID删除注册码。

输入示例

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

{
"RegisterCodeIds": [
"c24dcbca-4e83-463c-b1ae-f6a432c009f0",
"f86ccb49-520a-41bd-ba00-140a9dbed79f"
]
}
```

输出示例

```
{
"Response": {
"RequestId": "c08c4975-9dbc-4ac1-8a88-6943ffbde1d5"
}
}
```

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
InvalidParameterValue.InvalidRegisterCodeId	无效的注册码ID。
InvalidParameterValue.ParameterKeyDuplicated	参数 Key 重复。
MissingParameter	缺少参数错误。
ResourceNotFound.RegisterCodesNotFoundCode	查询不到注册码。
ResourceNotFound.RoleNotFound	角色不存在。
UnauthorizedOperation.AssumeRoleUnauthorized	角色扮演未授权。

查询注册码

最近更新时间：2025-04-28 01:58:55

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

接口用于查询注册码信息。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeRegisterCodes。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
RegisterCodeIds.N	否	Array of String	注册码ID。 每次请求的上限为 100。 参数不支持同时指定 `RegisterCodeIds` 和 `Filters` 。 示例值：["d0b7xxxx-a6xx-40x9-898x-44c9f508axxx"]
Offset	否	Integer	偏移量，默认为 0。 示例值：0
Limit	否	Integer	返回数量，默认为 20，最大值为 100。 示例值：10

3. 输出参数

参数名称	类型	描述
TotalCount	Integer	查询到的注册码总数。 示例值：10
RegisterCodeSet	Array of RegisterCodeInfo	注册码信息列表。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId ）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 根据注册码ID查询注册码

根据注册码ID查询注册码。

输入示例

```
POST / HTTP/1.1
Host: tat.tencentcloudapi.com
```

```
Content-Type: application/json
X-TC-Action: DescribeRegisterCodes
<公共请求参数>

{
"RegisterCodeIds": [
"8cca2d3b-7ac3-422a-98f0-8a5bc17cdc38"
],
"Offset": 0,
"Limit": 1
}
```

输出示例

```
{
"Response": {
"RegisterCodeSet": [
{
"CreatedTime": "2024-10-26T08:50:45Z",
"Description": "HAI instance register code, attach hai-8sclmptu",
"Enabled": false,
"ExpiredTime": "2024-10-26T12:50:45Z",
"InstanceNamePrefix": "HAI-Instance-hai-8sclmptu",
"IpAddressRange": "",
"RegisterCodeId": "8cca2d3b-7ac3-422a-98f0-8a5bc17cdc38",
"RegisterLimit": 10,
"RegisteredCount": 1,
"UpdatedTime": "2024-10-26T08:50:45Z"
}
],
"RequestId": "e0f011ac-6949-4726-a7d6-b28540f9d729",
"TotalCount": 1
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameterValue.InvalidFilter	Filter 无效。
InvalidParameterValue.InvalidRegisterCodeId	无效的注册码ID。
InvalidParameterValue.TooLong	长度超过限制。
ResourceNotFound.RoleNotFound	角色不存在。
UnauthorizedOperation.AssumeRoleUnauthorized	角色扮演未授权。

修改托管实例

最近更新时间：2025-04-25 01:54:41

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

接口用于修改托管实例信息。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：ModifyRegisterInstance。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
InstanceId	是	String	托管实例ID。 可通过 DescribeRegisterInstances(查询托管实例) 接口获取。 示例值：rins-8dz6cwxx
InstanceName	是	String	实例名称。有效长度为 1~60 字符。 示例值：register-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 修改托管实例

修改托管实例名为"abc"

输入示例

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

{
"InstanceId": "rins-xfjkunbgax",
"InstanceName": "webserver-01"
}
```

输出示例

```
{
"Response": {
"RequestId": "e0f011ac-6949-4726-a7d6-b28540f9d729"
}
}
```

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
InvalidParameterValue	参数取值错误。
InvalidParameterValue.InvalidRegisterInstanceId	无效的托管实例ID。
InvalidParameterValue.TooLong	长度超过限制。
MissingParameter	缺少参数错误。
ResourceNotFound.RoleNotFound	角色不存在。
UnauthorizedOperation.AssumeRoleUnauthorized	角色扮演未授权。

删除托管实例

最近更新时间：2025-04-25 01:54:42

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

接口用于删除托管实例。

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：DeleteRegisterInstance。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
InstanceId	是	String	托管实例ID。 可通过 DescribeRegisterInstances(查询托管实例) 接口获取。 示例值：["rins-8d5cxxxx"]

3. 输出参数

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

4. 示例

示例1 删除托管实例

根据实例ID删除托管实例。

输入示例

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

{
"InstanceId": "rinst-xdfaxfgax"
}
```

输出示例

```
{
"Response": {
"RequestId": "f86ccb49-520a-41bd-ba00-140a9dbed79f"
}
}
```

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
InvalidParameterValue.InvalidRegisterInstanceId	无效的托管实例ID。
MissingParameter	缺少参数错误。
ResourceNotFound.RegisterInstanceNotFoundCode	查询不到注册实例。
ResourceNotFound.RoleNotFound	角色不存在。
UnauthorizedOperation.AssumeRoleUnauthorized	角色扮演未授权。

查询托管实例

最近更新时间：2025-04-28 01:58:55

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

接口用于查询被托管的实例信息。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeRegisterInstances。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
InstanceIds.N	否	Array of String	托管实例 id。 每次请求的上限为 100。 参数不支持同时指定 `InstanceIds` 和 `Filters` 。 示例值：["rins-8d88xxxx"]
Filters.N	否	Array of Filter	过滤器列表。每次请求的 `Filters` 的上限为 10，`Filter.Values` 的上限为 5。参数不支持同时指定 `InstanceIds` 和 `Filters` 。 - instance-name 按照【托管实例名称】进行过滤。 类型：String 必选：否 - instance-id 按照【托管实例ID】进行过滤。 类型：String 必选：否 - register-code-id 按照【托管实例注册码ID】进行过滤。可通过 DescribeRegisterCodes(查询注册码) 接口获取。 类型：String 必选：否 - sys-name 按照【操作系统类型】进行过滤，取值：Linux | Windows。 类型：String 必选：否 - tag-key

参数名称	必选	类型	描述
			按照【标签键】进行过滤。 类型：String 必选：否 - tag-value 按照【标签值】进行过滤。 类型：String 必选：否 - tag:tag-key 按照【标签键值对】进行过滤。tag-key使用具体的标签键进行替换。 类型：String 必选：否 例如 Filter 为 {"Name": "tag:key1", "Values": ["v1", "v2"] }，即查询所有标签为 key1:v1 或 key1:v2 的资源。 示例值：[{"name":"register-code-id","value":"67c6df82-fa3e-48fb-9c17-07e3e1e9c2c3"}]
Offset	否	Integer	偏移量，默认为 0。 示例值：0
Limit	否	Integer	返回数量，默认为 20，最大值为 100。 示例值：10

3. 输出参数

参数名称	类型	描述
TotalCount	Integer	该实例注册过的注册码总数。 示例值：10
RegisterInstanceSet	Array of RegisterInstanceInfo	被托管的实例信息的列表。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 根据Filters查询托管实例

根据Filters查询托管实例。

输入示例

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

{
"Filters": [
{
"Name": "instance-name",
"Values": [
"WebServer-01"
]
```

```
}
],
"Offset": 0,
"Limit": 1
}
```

输出示例

```
{
"Response": {
"TotalCount": 1,
"RegisterInstanceSet": [
{
"RegisterCodeId": "d0b7xxxx-a6xx-40x9-898x-44c9f508axxx",
"InstanceId": "rins-uf673dgi",
"InstanceName": "WebServer-01",
"MachineId": "22c4dvvv-3ae9-4ff3-8331-d44147c96f8b",
"SystemName": "Windows",
"HostName": "webserver01",
"LocalIp": "10.0.0.1",
"PublicKey": "-----BEGIN RSA PUBLIC KEY-----\nAAAAAAAAAQEA1vVOon7U12dLFl7AOZjGnfWSMJ4rhHAagLne85Qbyn7rrow90bZF\ngsO9cpUfljliymtGAe1ZxkZZAWbwYeiS+x3KMO3rsbaXZtb9CiPSuMmyCcoyNRWy\nztLhIAHz6TFdoYYPdepY/HrL1rppGaVvUT6ufZrWIw2wF4KzPsIwGmfKYDrP+JOd\nfxkdewMwaFUP8xotqF+qi6YVbBMYSEGQyU9n42FTNxHMsLrBf1yPIcyaD+itWbWk\nyG0KcHTF2gbtgHqGV9wkc5jcqJYK3iJ1bQgEIWZ1lThRncOnHKHmSEnQ/XD5+6Hi\nLdVILVm0Gu8cnelE34kTcIqUAIQcB9wJBBBBBBBB\n-----END RSA PUBLIC KEY-----\n",
"Status": "Online",
"CreatedTime": "2020-09-22T08:00:00Z",
"UpdatedTime": "2020-09-22T08:00:00Z"
}
],
"RequestId": "e28b9a8c-d7d3-4a12-bf8a-0123456789ab"
}
}
```

示例2 根据托管实例ID查询托管实例

根据托管实例ID查询托管实例。

输入示例

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

{
"InstanceIds": [
"rins-uf673dgi"
],
"Offset": 0,
"Limit": 1
}
```

输出示例

```
{
"Response": {
"TotalCount": 1,
"RegisterInstanceSet": [
{
"RegisterCodeId": "d0b7xxxx-a6xx-40x9-898x-44c9f508axxx",
"InstanceId": "rins-uf673dgi",
"InstanceName": "WebServer-01",
"MachineId": "22c4dvvv-3ae9-4ff3-8331-d44147c96f8b",
"SystemName": "Windows",
"HostName": "webserver01",
"LocalIp": "10.0.0.1",
"PublicKey": "-----BEGIN RSA PUBLIC KEY-----\nAAAAAAAAAQEA1vVOon7U12dLFl7AOZjGnfWSMJ4rhHAagLne85Qbyn7rrow90bZF\ngsO9cpUfljliymtGAe1ZxkZZAWbwYeiS+x3KMO3rsbaXZtb9CiPSuMmyCcoyNRWy\nztLhIAHz6TFdoYYPdepY/HrL1rppGaVvUT6ufZrWIw2wF4KzPsIwGmfKYDrP+JOd\nfxkdewMwaFUP8xotqF+qi6YVbBMYSEGQyU9n42FTNxHMsLrBf1yPIcyaD+itWbWk\nyG0KcHTF2gbtgHqGV9wkc5jcqJYK3iJ1bQgEIWZ1lThRncOnHKHmSEnQ/XD5+6Hi\nLdVILVm0Gu8cnelE34kTcIqUAIQcB9wJBBBBBBBB\n-----END RSA PUBLIC KEY-----\n",
"Status": "Online",
"CreatedTime": "2020-09-22T08:00:00Z",
"UpdatedTime": "2020-09-22T08:00:00Z"
}
],
"RequestId": "e28b9a8c-d7d3-4a12-bf8a-0123456789ab"
}
}
```


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
InvalidParameterValue.InvalidFilter	Filter 无效。
InvalidParameterValue.InvalidRegisterInstanceId	无效的托管实例ID。

错误码	描述
ResourceNotFound.RoleNotFound	角色不存在。
UnauthorizedOperation.AssumeRoleUnauthorized	角色扮演未授权。

场景相关接口

查询场景

最近更新时间：2025-04-28 01:58:57

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于查询场景详情。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeScenes。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
SceneIds.N	否	Array of String	场景 ID 数组。 每次请求的上限为 100。 参数不支持同时指定 SceneIds 和 Filters 。 示例值：["sc-12345678"]
Filters.N	否	Array of Filter	过滤条件。 - scene-id - String - 是否必填：否 -（过滤条件）按照场景 ID 过滤。 - scene-name - String - 是否必填：否 -（过滤条件）按照场景名称过滤。 - created-by - String - 是否必填：否 -（过滤条件）按照场景创建者过滤，目前仅支持 TAT，代表公共场景。 每次请求的 Filters 的上限为10， Filter.Values 的上限为5。参数不支持同时指定 SceneIds 和 Filters 。 示例值：{"scene-id": "sc-12345678"}
Limit	否	Integer	返回数量，默认为20，最大值为100。关于 Limit 的更进一步介绍请参考 API 简介中的相关小节。 示例值：20
Offset	否	Integer	偏移量，默认为0。关于 Offset 的更进一步介绍请参考 API 简介中的相关小节。 示例值：0

3. 输出参数

参数名称	类型	描述
TotalCount	Integer	符合条件的场景总数。 示例值：1
SceneSet	Array of Scene	场景详情列表。 示例值：[{ "CreatedBy": "TAT", "CreatedTime": "2024-09-23T07:52:26Z", "SceneId": "sc-r8ct2mkc", "SceneName": "Windows", "UpdatedTime": "2024-09-23T08:12:27Z" }]
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId ）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询场景

输入示例

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

{
"SceneIds": [
"sc-12345678"
],
"Offset": 0,
"Limit": 20
}
```

输出示例

```
{
"Response": {
"RequestId": "eb973a12-71e3-4c0c-b1d8-4b863e5f5daf",
"TotalCount": 1,
"SceneSet": [
{
"SceneId": "sc-12345678",
"SceneName": "运维场景",
"CreatedBy": "TAT",
"CreatedTime": "2020-11-02T02:48:11+00:00",
"UpdatedTime": "2020-11-02T02:48:11+00:00"
}
]
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError	内部错误。
InvalidFilter	无效的过滤器
InvalidParameter	参数错误。
InvalidParameter.ConflictParameter	参数冲突。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.InvalidFilter	Filter 无效。
InvalidParameterValue.InvalidSceneId	无效的场景 ID。
InvalidParameterValue.InvalidSceneName	无效的场景名称。
LimitExceeded.FilterValueExceeded	填写的 Filter 取值过多。
UnauthorizedOperation.CamAuthFailed	CAM鉴权失败。
UnauthorizedOperation.MFAExpired	Multi-Factor Authentication(MFA) 过期。
UnauthorizedOperation.MFANotFound	Multi-Factor Authentication(MFA) 不存在。

统计相关接口
查询地域列表

最近更新时间：2025-04-25 01:54:41

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于查询 TAT 产品后台地域列表。
RegionState 为 AVAILABLE，代表该地域的 TAT 后台服务已经可用；未返回，代表该地域的 TAT 后台服务尚不可用。

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：DescribeRegions。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。

3. 输出参数

参数名称	类型	描述
TotalCount	Integer	地域数量 示例值：1
RegionSet	Array of RegionInfo	地域信息列表 示例值：[{ "Region": "ap-guangzhou", "RegionName": "广州", "RegionState": "AVAILABLE" }]
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。 定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询地域列表

此接口用于查询 TAT 产品后台地域列表。 RegionState 为 AVAILABLE，代表该地域的 TAT 后台服务已经可用；未返回，代表该地域的 TAT 后台服务尚不可用

输入示例

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

{}
```

输出示例

```
{
"Response": {
"TotalCount": 12,
"RegionSet": [
{
"Region": "ap-guangzhou",
"RegionName": "广州",
"RegionState": "AVAILABLE"
},
{
"Region": "ap-nanjing",
"RegionName": "南京",
"RegionState": "AVAILABLE"
},
{
"Region": "ap-shanghai",
"RegionName": "上海",
"RegionState": "AVAILABLE"
},
{
"Region": "ap-hongkong",
"RegionName": "中国香港",
"RegionState": "AVAILABLE"
},
{
"Region": "ap-beijing",
"RegionName": "北京",
"RegionState": "AVAILABLE"
},
{
"Region": "ap-singapore",
"RegionName": "新加坡",
"RegionState": "AVAILABLE"
},
{
"Region": "na-siliconvalley",
"RegionName": "硅谷",
"RegionState": "AVAILABLE"
},
{
"Region": "ap-chengdu",
"RegionName": "成都",
"RegionState": "AVAILABLE"
},
{
"Region": "eu-frankfurt",
"RegionName": "法兰克福",
"RegionState": "AVAILABLE"
},
{
"Region": "ap-seoul",
"RegionName": "首尔",
"RegionState": "AVAILABLE"
},
{
"Region": "ap-chongqing",
```

```
"RegionName": "重庆",
"RegionState": "AVAILABLE"
}
],
"RequestId": "6fb7f9db-b7da-4cb8-a912-3a3b1690f3a6"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError	内部错误。
UnauthorizedOperation.CamAuthFailed	CAM鉴权失败。
UnauthorizedOperation.InvalidToken	Token 无效。
UnauthorizedOperation.MFAExpired	Multi-Factor Authentication(MFA) 过期。
UnauthorizedOperation.MFANotFound	Multi-Factor Authentication(MFA) 不存在。

获取配额信息

最近更新时间：2025-04-25 01:54:41

1. 接口描述

接口请求域名： tat.tencentcloudapi.com 。

此接口用于获取配额信息

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeQuotas。
Version	是	String	公共参数，本接口取值：2020-10-28。
Region	是	String	公共参数，详见产品支持的 地域列表。
ResourceNames.N	是	Array of String	资源名称 取值为： - COMMAND：命令 - REGISTER_CODE：托管实例注册码 示例值：["COMMAND"]

3. 输出参数

参数名称	类型	描述
GeneralResourceQuotaSet	Array of GeneralResourceQuotaSet	资源额度列表 示例值：[{ "ResourceName": "COMMAND", "ResourceQuotaTotal": 500, "ResourceQuotaUsed": 27 }]
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId ）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询"COMMAND","REGISTER_CODE"资源

输入示例

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

{
"ResourceNames": [
"COMMAND",
```

```
"REGISTER_CODE"
]
}
```

输出示例

```
{
"Response": {
"RequestId": "321a2144-8e11-419e-b33c-6f6bfc25be4d",
"GeneralResourceQuotaSet": [
{
"ResourceName": "COMMAND",
"ResourceQuotaUsed": 26,
"ResourceQuotaTotal": 509
},
{
"ResourceName": "REGISTER_CODE",
"ResourceQuotaUsed": 12,
"ResourceQuotaTotal": 5000
}
]
}
}
```

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
MissingParameter	缺少参数错误。
ResourceUnavailable.InvalidResourceQuotaResourceCode	不存在的资源名称

数据结构

最近更新时间：2025-04-28 01:58:57

AutomationAgentInfo

自动化助手客户端信息

被如下接口引用：DescribeAutomationAgentStatus。

名称	类型	描述
InstanceId	String	实例ID。 示例值：ins-tatxxxyyy
Version	String	Agent 版本号。 示例值：1.0.19
LastHeartbeatTime	Timestamp ISO8601	上次心跳时间 示例值：2023-12-11T07:45:27Z
AgentStatus	String	Agent状态，取值范围： Online：在线，Offline：离线 示例值：Online
Environment	String	Agent运行环境，取值范围：Linux：Linux实例Windows：Windows实例 示例值：Linux
SupportFeatures	Array of String	Agent 支持的功能列表。 示例值：["NEW_FEATURE"]

Command

命令详情。

被如下接口引用：DescribeCommands。

名称	类型	描述
CommandId	String	命令ID。 示例值：cmd-aaaaaaaa
CommandName	String	命令名称。 示例值：cmdname
Description	String	命令描述。 示例值：the cmd
Content	String	Base64编码后的命令内容。 示例值：bHMgMTIzCmVjaG8ge3tifX0ge3tjfX0=
CommandType	String	命令类型。取值为 SHELL、POWERSHELL、BAT 之一。 示例值：SHELL
WorkingDirectory	String	命令执行路径。 示例值：/root
Timeout	Integer	命令超时时间。 示例值：60
CreatedTime	Timestamp ISO8601	命令创建时间。格式为：YYYY-MM-DDThh:mm:ssZ 示例值：2021-05-12T02:49:04Z
UpdatedTime	Timestamp ISO8601	命令更新时间。格式为：YYYY-MM-DDThh:mm:ssZ 示例值：2021-05-12T02:49:04Z
EnableParameter	Boolean	是否启用自定义参数功能。 示例值：false

名称	类型	描述
DefaultParameters	String	自定义参数的默认取值。 示例值：{"varA": "222"}
DefaultParameterConfs	Array of DefaultParameterConf	自定义参数的默认取值。 示例值： [{"ParameterName":"test01","ParameterValue":"12345","ParameterDescription":"for test01"}]
Scenes	Array of String	命令关联的场景 示例值：["sc-12345678"]
FormattedDescription	String	命令的结构化描述。公共命令有值，用户命令为空字符串。 示例值：{"scenes":["sc-e22tn6f2"],"cmd_description":{"en":"english description","zh-cn":"中文描述","name-en":"EngName","name-zh-cn":"中文名"},"parameter_description":{},"cmdName":"name","cmdDesc":"desc"}
CreatedBy	String	命令创建者。TAT 代表公共命令，USER 代表个人命令。 示例值：TAT
Tags	Array of Tag	命令关联的标签列表。 示例值：[{"Key":"k", "Value":"v"}]
Username	String	在实例上执行命令的用户名。 示例值：root
OutputCOSBucketUrl	String	日志上传的cos bucket 地址。 示例值：https://BucketName-123454321.cos.ap-beijing.myqcloud.com
OutputCOSKeyPrefix	String	日志在cos bucket中的目录。 示例值：aa/bb/cc

CommandDocument

命令执行详情。

被如下接口引用：DescribeInvocationTasks。

名称	类型	描述
Content	String	Base64 编码后的执行命令。 示例值：cHdk
CommandType	String	命令类型。取值为 SHELL、POWERSHELL、BAT 之一。 示例值：SHELL
Timeout	Integer	超时时间。单位：秒。 示例值：60
WorkingDirectory	String	执行路径。 示例值：/root
Username	String	执行用户。 示例值：root
OutputCOSBucketUrl	String	保存输出的 COS Bucket 链接。 示例值：https://<名称>-<AppId>.cos.ap-beijing.myqcloud.com
OutputCOSKeyPrefix	String	保存输出的文件名称前缀。 示例值：agent

DefaultParameterConf

自定义参数。

被如下接口引用：CreateCommand, DescribeCommands, ModifyCommand, RunCommand。

名称	类型	必选	描述
ParameterName	String	是	参数名。 示例值：name
ParameterValue	String	是	参数默认值。 示例值：value
ParameterDescription	String	否	参数描述。 示例值：This is a parameter.

Filter

描述键值对过滤器，用于条件过滤查询。例如过滤ID、名称、状态等

- **若存在多个`Filter`时，`Filter`间的关系为逻辑与（`AND`）关系。**
- **若同一个`Filter`存在多个`Values`，同一`Filter`下`Values`间的关系为逻辑或（`OR`）关系。**

以DescribeCommands接口的`Filters`为例。若我们需要查询命令名称（`command-name`）为“打印工作目录”*并且*命令类型（`command-type`）为“POWERSHELL”*或者*“BAT”时，可如下实现：

```
Filters.0.Name=command-name
&Filters.0.Values.0=打印工作目录

&Filters.1.Name=command-type
&Filters.1.Values.0=POWERSHELL
&Filters.1.Values.1=BAT
```

被如下接口引用：DescribeAutomationAgentStatus, DescribeCommands, DescribeInvocationTasks, DescribeInvocations, DescribeInvokers, DescribeRegisterInstances, DescribeScenes。

名称	类型	必选	描述
Name	String	是	需要过滤的字段。 示例值：command-type
Values	Array of String	是	字段的过滤值。 示例值：["POWERSHELL", "BAT"]

GeneralResourceQuotaSet

用户配额信息。

被如下接口引用：DescribeQuotas。

名称	类型	必选	描述
ResourceName	String	否	资源名称 取值为： - COMMAND：命令 - REGISTER_CODE：托管实例注册码 示例值：COMMAND
ResourceQuotaUsed	Integer	否	已使用额度 示例值：200
ResourceQuotaTotal	Integer	否	总额度 示例值：500

Invocation

执行活动详情。

被如下接口引用：DescribeInvocations。

名称	类型	描述
InvocationId	String	执行活动ID。 示例值：inv-xxxxxxxx
CommandId	String	命令ID。 示例值：cmd-xxxxxxxx
InvocationStatus	String	执行任务状态。取值范围： - PENDING：等待下发 - RUNNING：命令运行中 - CANCELLING：取消中 - SUCCESS：命令成功 - TIMEOUT：命令超时 - FAILED：命令失败 - CANCELLED：命令全部取消 - PARTIAL_FAILED：命令部分失败 - PARTIAL_CANCELLED：命令部分取消 示例值：SUCCESS
InvocationTaskBasicInfoSet	Array of InvocationTaskBasicInfo	执行任务信息列表。 示例值：[{"InstanceId":"lhins-71itjmdd","InvocationTaskId":"invt-bd1w17gx22","TaskStatus":"SUCCESS"}]
Description	String	执行活动描述。 示例值：Test Invocation
StartTime	Timestamp ISO8601	执行活动开始时间。格式为：YYYY-MM-DDThh:mm:ssZ 示例值：2025-03-05T20:07:29Z
EndTime	Timestamp ISO8601	执行活动结束时间。格式为：YYYY-MM-DDThh:mm:ssZ 示例值：2025-03-05T20:07:29Z
CreatedTime	Timestamp ISO8601	执行活动创建时间。格式为：YYYY-MM-DDThh:mm:ssZ 示例值：2025-03-05T20:07:29Z
UpdatedTime	Timestamp ISO8601	执行活动更新时间。格式为：YYYY-MM-DDThh:mm:ssZ 示例值：2025-03-05T20:07:29Z
Parameters	String	自定义参数取值。 示例值：{"varA": "222"}
DefaultParameters	String	自定义参数的默认取值。 示例值：{"varA": "222"}
InstanceKind	String	执行命令的实例类型，取值范围：CVM、LIGHTHOUSE。 示例值：CVM
Username	String	在实例上执行命令时使用的用户名。 示例值：root
InvocationSource	String	调用来源。 - USER：来源于用户调用。 - INVOKER：来源于定时执行。 示例值：USER
CommandContent	String	base64编码的命令内容 示例值：cHdk
CommandType	String	命令类型 示例值：SHELL
Timeout	Integer	执行命令过期时间， 单位秒 示例值：60

名称	类型	描述
WorkingDirectory	String	执行命令的工作路径 示例值：/root
OutputCOSBucketUrl	String	日志上传的cos bucket 地址。 示例值：https://BucketName-123454321.cos.ap-beijing.myqcloud.com
OutputCOSKeyPrefix	String	日志在cos bucket中的目录。 示例值：aa/bb/cc

InvocationTask

执行任务。

被如下接口引用：DescribeInvocationTasks。

名称	类型	描述
InvocationId	String	执行活动ID。 示例值：inv-xxxxxxxx
InvocationTaskId	String	执行任务ID。 示例值：invt-xxxxxxxx
CommandId	String	命令ID。 示例值：cmd-xxxxxxxx
TaskStatus	String	执行任务状态。取值范围： - PENDING：等待下发 - DELIVERING：下发中 - DELIVER_DELAYED：延时下发 - DELIVER_FAILED：下发失败 - START_FAILED：命令启动失败 - RUNNING：命令运行中 - SUCCESS：命令成功 - FAILED：命令执行失败，执行完退出码不为 0 - TIMEOUT：命令超时 - TASK_TIMEOUT：客户端无响应 - CANCELLING：取消中 - CANCELLED：已取消（命令启动前就被取消） - TERMINATED：已中止（命令执行期间被取消） 示例值：SUCCESS
InstanceId	String	实例ID。 示例值：ins-xxxxxxxx
TaskResult	TaskResult	执行结果。 示例值：{"Dropped":0,"ExecEndTime":"2024-11-13T02:10:32Z","ExecStartTime":"2024-11-13T02:10:32Z","ExitCode":7,"Output":"","OutputUploadCOSErrorInfo":"","OutputUrl":""}
StartTime	Timestamp ISO8601	执行任务开始时间。格式为：YYYY-MM-DDThh:mm:ssZ 注意：此字段可能返回 null，表示取不到有效值。 示例值：2025-03-05T20:07:29Z
EndTime	Timestamp ISO8601	执行任务结束时间。格式为：YYYY-MM-DDThh:mm:ssZ 注意：此字段可能返回 null，表示取不到有效值。 示例值：2025-03-05T20:07:29Z
CreatedTime	Timestamp ISO8601	创建时间。格式为：YYYY-MM-DDThh:mm:ssZ 示例值：2025-03-05T20:07:29Z
UpdatedTime	Timestamp ISO8601	更新时间。格式为：YYYY-MM-DDThh:mm:ssZ 示例值：2025-03-05T20:07:29Z
CommandDocument	CommandDocument	执行任务所执行的命令详情。 示例值： {"CommandType":"SHELL","Content":"cHdk","OutputCOSBucketUrl":"","OutputCOSKeyPre

名称	类型	描述
ErrorInfo	String	执行任务失败时的错误信息。 示例值：working_directory not exists
InvocationSource	String	调用来源。 - USER：来源于用户调用。 - INVOKER：来源于定时执行。 示例值：USER

InvocationTaskBasicInfo

执行活动任务简介。

被如下接口引用：DescribeInvocations。

名称	类型	描述
InvocationTaskId	String	执行任务ID。 示例值：invt-aaaabbbb
TaskStatus	String	执行任务状态。取值范围： - PENDING：等待下发 - DELIVERING：下发中 - DELIVER_DELAYED：延时下发 - DELIVER_FAILED：下发失败 - START_FAILED：命令启动失败 - RUNNING：命令运行中 - SUCCESS：命令成功 - FAILED：命令执行失败，执行完退出码不为 0 - TIMEOUT：命令超时 - TASK_TIMEOUT：客户端无响应 - CANCELLING：取消中 - CANCELLED：已取消（命令启动前就被取消） - TERMINATED：已中止（命令执行期间被取消） 示例值：SUCCESS
InstanceId	String	实例ID。 示例值：ins-aaaabbbb

Invoker

执行器信息。

被如下接口引用：DescribeInvokers。

名称	类型	描述
InvokerId	String	执行器ID。 示例值：ivk-27yagap9
Name	String	执行器名称。 示例值：invoker-test
Type	String	执行器类型。目前仅支持 SCHEDULE 一种。 示例值：SCHEDULE
CommandId	String	命令ID。 示例值：cmd-m7uma92n
Username	String	用户名。 示例值：root
Parameters	String	自定义参数。 示例值：{"var": "1"}

名称	类型	描述
InstanceIds	Array of String	实例ID列表。 示例值：["ins-yx05ky8j"]
Enable	Boolean	执行器是否启用。 示例值：true
ScheduleSettings	ScheduleSettings	执行器周期计划。周期执行器会返回此字段。
CreatedTime	Timestamp ISO8601	创建时间。格式为：YYYY-MM-DDThh:mm:ssZ 示例值：2021-08-30T06:42:02Z
UpdatedTime	Timestamp ISO8601	修改时间。格式为：YYYY-MM-DDThh:mm:ssZ 示例值：2021-08-30T06:42:02Z

InvokerRecord

执行器执行记录。

被如下接口引用：DescribeInvokerRecords。

名称	类型	描述
InvokerId	String	执行器ID。 示例值：ivk-b7s3qa5l
InvokeTime	Timestamp ISO8601	执行时间。格式为：YYYY-MM-DDThh:mm:ssZ 示例值：2021-10-30T00:00:00Z
Reason	String	执行原因。 示例值：start an invocation at scheduled time.
InvocationId	String	命令执行ID。 示例值：inv-4ybg8gmj
Result	String	触发结果。 - PENDING：等待下发 - RUNNING：命令运行中 - CANCELLING：取消中 - SUCCESS：命令成功 - TIMEOUT：命令超时 - FAILED：命令失败 - CANCELLED：命令全部取消 - PARTIAL_FAILED：命令部分失败 - PARTIAL_CANCELLED：命令部分取消 示例值：SUCCESS

RegionInfo

描述单个地域信息

被如下接口引用：DescribeRegions。

名称	类型	描述
Region	String	地域名称，例如，ap-guangzhou 示例值：ap-guangzhou
RegionName	String	地域描述，例如: 广州 示例值：广州
RegionState	String	地域是否可用状态，AVAILABLE 代表可用，UNAVAILABLE 代表不可用。 示例值：AVAILABLE

RegisterCodeInfo

注册码信息。

被如下接口引用：DescribeRegisterCodes。

名称	类型	描述
RegisterCodeId	String	注册码ID。 示例值：d0b7xxxx-a6xx-40x9-898x-44c9f508axxx
Description	String	注册码描述。 示例值：此注册码用于注册托管实例。
InstanceNamePrefix	String	注册实例名称前缀。 示例值：register-test
RegisterLimit	Integer	该注册码允许注册的实例数目。 示例值：100
ExpiredTime	Timestamp ISO8601	该注册码的过期时间，按照 ISO8601 标准表示，并且使用 UTC 时间。 格式为： YYYY-MM-DDThh:mm:ssZ。 注意：此字段可能返回 null，表示取不到有效值。 示例值：2023-09-07T13:13:29Z
IpAddressRange	String	该注册码限制tat_agent只能从IpAddressRange所描述公网出口进行注册。 示例值：133.12.234.0/24
Enabled	Boolean	该注册码是否可用。 示例值：true
RegisteredCount	Integer	该注册码已注册数目。 示例值：10
CreatedTime	Timestamp ISO8601	注册码创建时间，按照 ISO8601 标准表示，并且使用 UTC 时间。 格式为： YYYY-MM-DDThh:mm:ssZ。 注意：此字段可能返回 null，表示取不到有效值。 示例值：2023-12-01 00:00:00
UpdatedTime	Timestamp ISO8601	注册码最近一次更新时间，按照 ISO8601 标准表示，并且使用 UTC 时间。 格式为： YYYY-MM-DDThh:mm:ssZ。 注意：此字段可能返回 null，表示取不到有效值。 示例值：2023-12-01 00:00:00

RegisterInstanceInfo

注册实例信息。

被如下接口引用：DescribeRegisterInstances。

名称	类型	描述
RegisterCodeId	String	注册码ID。 示例值：d0b7xxxx-a6xx-40x9-898x-44c9f508axxx
InstanceId	String	托管实例ID。 示例值：rins-8d5cxxxx
InstanceName	String	托管实例名。 示例值：register-test
MachineId	String	机器ID。 示例值：00xxxxx-0fxx-xxxx-xxxx-33951xxxxxxx
SystemName	String	系统名。取值：Linux | Windows。 示例值：Linux
HostName	String	主机名。 示例值：VM-0-01-ubuntu
LocalIp	String	内网IP。 示例值：10.0.0.1

名称	类型	描述
PublicKey	String	公钥。 示例值：-----BEGIN RSA PUBLIC KEY-----\nXXXXX...
Status	String	托管状态。 返回Online表示实例正在托管，返回Offline表示实例未托管。 示例值：Online
CreatedTime	Timestamp ISO8601	创建时间。格式为：YYYY-MM-DDThh:mm:ssZ 示例值：2025-03-05T20:07:29Z
UpdatedTime	Timestamp ISO8601	上次更新时间。格式为：YYYY-MM-DDThh:mm:ssZ 示例值：2025-03-05T20:07:29Z
Tags	Array of Tag	标签 示例值：[{"name": "owner", "value": "zhang san"}]

Scene

场景详情。

被如下接口引用：DescribeScenes。

名称	类型	必选	描述
SceneId	String	否	场景 ID 。 示例值：sc-12345678
SceneName	String	否	场景名称。 示例值：运维场景
CreatedBy	String	否	场景创建者。 - TAT：公共场景 示例值：TAT
CreatedTime	String	否	创建时间。格式为：YYYY-MM-DDThh:mm:ssZ 示例值：2021-05-12T02:49:04Z
UpdatedTime	String	否	更新时间。格式为：YYYY-MM-DDThh:mm:ssZ 示例值：2021-05-12T02:49:04Z

ScheduleSettings

周期执行器设置。

被如下接口引用：CreateInvoker, DescribeInvokers, ModifyInvoker。

名称	类型	必选	描述
Policy	String	是	执行策略： - ONCE：单次执行 - RECURRENCE：周期执行 示例值：ONCE
Recurrence	String	否	触发 Crontab 表达式。Policy 为 RECURRENCE 时，需要指定此字段。Crontab 按北京时间解析。 示例值：0 0 1 * *
InvokeTime	Timestamp ISO8601	否	执行器下次执行时间。Policy 为 ONCE 时，需要指定此字段。 时间格式为：YYYY-MM-DDThh:mm:ssZ 示例值：2021-09-01T00:00:00+08:00

Tag

标签

被如下接口引用：CreateCommand, DescribeCommands, DescribeRegisterInstances, RunCommand。

名称	类型	必选	描述
Key	String	是	标签键。 示例值：tag-key
Value	String	是	标签值。 示例值：tag-value

TaskResult

任务结果。

被如下接口引用：DescribeInvocationTasks。

名称	类型	描述
ExitCode	Integer	命令执行ExitCode。 示例值：0
Output	String	Base64编码后的命令输出。最大长度24KB。 示例值：aGVsbG8sd29ybGQ=
ExecStartTime	Timestamp ISO8601	命令执行开始时间。格式为：YYYY-MM-DDThh:mm:ssZ 注意：此字段可能返回 null，表示取不到有效值。 示例值：2025-03-05T20:07:29Z
ExecEndTime	Timestamp ISO8601	命令执行结束时间。格式为：YYYY-MM-DDThh:mm:ssZ 注意：此字段可能返回 null，表示取不到有效值。 示例值：2025-03-05T20:07:29Z
Dropped	Integer	命令最终输出被截断的字节数。 示例值：100
OutputUrl	String	日志在cos中的地址 示例值：https://BucketName-123454321.cos.ap-beijing.myqcloud.com
OutputUploadCOSErrorInfo	String	日志上传cos的错误信息。 示例值：Failed to upload output to cos

错误码

最近更新时间：2025-06-11 01:53:11

功能说明

如果返回结果中存在 Error 字段，则表示调用 API 接口失败。例如：

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

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

错误码列表

公共错误码

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

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
AuthFailure	CAM签名/鉴权错误。
FailedOperation.CVMError	调用 CVM 失败。
FailedOperation.LighthouseError	调用 Lighthouse 失败。
InvalidFilter	无效的过滤器
InvalidParameter.ConflictParameter	参数冲突。
InvalidParameter.InvalidUsername	无效用户名。
InvalidParameter.ParameterNameDuplicated	参数名称重复。
InvalidParameterValue.AgentUnsupportedCommandType	Agent不支持此命令类型。
InvalidParameterValue.CommandContentInvalid	Command 内容无效。
InvalidParameterValue.CommandNameDuplicated	Command 名称重复。
InvalidParameterValue.ID	ID 格式错误。
InvalidParameterValue.InconsistentID	ID 数组中，ID 格式错误或格式不一致。
InvalidParameterValue.InconsistentInstance	实例类型不一致。
InvalidParameterValue.InstanceIsNotRelatedToInvocation	实例ID与执行活动无关。
InvalidParameterValue.InvalidCommandId	CommandId 无效。
InvalidParameterValue.InvalidCommandName	Command 名称无效。

错误码	说明
InvalidParameterValue.InvalidContent	命令内容无效。
InvalidParameterValue.InvalidCronExpression	Crontab 表达式无效。
InvalidParameterValue.InvalidFilter	Filter 无效。
InvalidParameterValue.InvalidInstanceId	实例ID无效。
InvalidParameterValue.InvalidInvocationId	不合法的执行活动ID。
InvalidParameterValue.InvalidInvocationTaskId	不合法的执行任务ID。
InvalidParameterValue.InvalidInvokerId	InvokerId 无效。
InvalidParameterValue.InvalidOutputCOSBucketUrl	OutputCOSBucketUrl 无效。
InvalidParameterValue.InvalidOutputCOSKeyPrefix	OutputCOSKeyPrefix 无效。
InvalidParameterValue.InvalidRegisterCodeId	无效的注册码ID。
InvalidParameterValue.InvalidRegisterInstanceId	无效的托管实例ID。
InvalidParameterValue.InvalidSceneId	无效的场景 ID。
InvalidParameterValue.InvalidSceneName	无效的场景名称。
InvalidParameterValue.InvalidTimeFormat	无效的时间格式。
InvalidParameterValue.InvalidUsername	用户名不合法。
InvalidParameterValue.InvalidWorkingDirectory	命令执行路径不合法。
InvalidParameterValue.InvokeTimeExpired	调用时间已过期。
InvalidParameterValue.LackOfParameterInfo	已启用自定义参数功能，但缺失自定义参数信息。
InvalidParameterValue.LackOfParameters	未提供 Parameters 信息。
InvalidParameterValue.LimitExceeded	超过参数限制。
InvalidParameterValue.ParameterDisabled	未启用自定义参数功能。
InvalidParameterValue.ParameterInvalidJsonFormat	参数为非法 json string 格式。
InvalidParameterValue.ParameterKeyContainsInvalidChar	参数 Key 包含非法字符。
InvalidParameterValue.ParameterKeyDuplicated	参数 Key 重复。
InvalidParameterValue.ParameterKeyLenExceeded	参数 Key 过长。
InvalidParameterValue.ParameterNumberExceeded	参数数目过多。
InvalidParameterValue.ParameterValueNotString	参数 Value 非 string 类型。
InvalidParameterValue.Range	参数取值范围不合法。
InvalidParameterValue.SupportParametersOnlyIfEnableParameter	未启用自定义参数功能。
InvalidParameterValue.TooLarge	参数取值过大。
InvalidParameterValue.TooLong	长度超过限制。
InvalidParameterValue.TooSmall	参数取值过小。
LimitExceeded.FilterValueExceeded	填写的 Filter 取值过多。
OperationDenied	操作被拒绝。
ResourceNotFound.CommandNotFound	命令不存在。
ResourceNotFound.InstanceNotFound	实例不存在。
ResourceNotFound.InvocationNotFound	执行活动未找到。

错误码	说明
ResourceNotFound.RegisterCodesNotFoundCode	查询不到注册码。
ResourceNotFound.RegisterInstanceNotFoundCode	查询不到注册实例。
ResourceNotFound.RoleNotFound	角色不存在。
ResourceUnavailable.AgentNotInstalled	Agent 未安装。
ResourceUnavailable.AgentStatusNotOnline	Agent 不在线。
ResourceUnavailable.CommandInExecuting	命令正在执行中。
ResourceUnavailable.CommandInInvoker	命令已关联执行器。
ResourceUnavailable.InstanceStateNotRunning	实例未处于运行中。
ResourceUnavailable.InvalidResourceQuotaResourceCode	不存在的资源名称
ResourceUnavailable.LighthouseUnsupportedRegion	Lighthouse 尚不支持指定的地域。
ResourceUnavailable.UserHasNoQuotaCode	用户配额已用完
UnauthorizedOperation.AssumeRoleUnauthorized	角色扮演未授权。
UnauthorizedOperation.CamAuthFailed	CAM鉴权失败。
UnauthorizedOperation.InvalidToken	Token 无效。
UnauthorizedOperation.MFAExpired	Multi-Factor Authentication(MFA) 过期。
UnauthorizedOperation.MFANotFound	Multi-Factor Authentication(MFA) 不存在。