云顾问 – 混沌演练

API 文档

产品文档

文档目录

API 文档
更新历史

最近更新时间：2026-07-10 16:48:55

第 3 次发布

发布时间：2026-07-10 16:47:32

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- CreateTaskFromAction
  - 新增入参：TaskTags
- ModifyTaskRunStatus
  - 新增入参：Issue, Record, IncludeRecordInReport

第 2 次发布

发布时间：2025-08-01 17:17:22

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- DescribeTaskList
  - 新增入参：ArchId, ArchName

第 1 次发布

发布时间：2024-09-19 09:52:28

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- CreateTaskFromAction
- CreateTaskFromTemplate
- DeleteTask
- DescribeActionFieldConfigList
- DescribeActionLibraryList
- DescribeObjectTypeList
- DescribeTask
- DescribeTaskExecuteLogs
- DescribeTaskList
- DescribeTaskPolicyTriggerLog
- DescribeTemplate
- DescribeTemplateList
- ExecuteTask
- ExecuteTaskInstance
- ModifyTaskRunStatus
- TriggerPolicy

新增数据结构：

- ActionFieldConfigDetail
- ActionFieldConfigResult
- ActionFilter
- ActionLibraryListResult
- ApmServiceInfo
- DescribePolicy
- ObjectType
- ObjectTypeConfig
- ObjectTypeConfigFields
- ObjectTypeJsonParse
- PolicyTriggerLog
- ResourceOffline
- TagWithCreate
- TagWithDescribe
- Task
- TaskConfig
- TaskGroup
- TaskGroupAction

- TaskGroupActionConfig
- TaskGroupConfig
- TaskGroupInstance
- TaskGroupInstancesExecuteRules
- TaskListItem
- TaskMonitor
- TaskReportInfo
- Template
- TemplateGroup
- TemplateGroupAction
- TemplateListItem
- TemplateMonitor
- TemplatePolicy

简介

最近更新时间：2026-07-10 16:48:49

概述

智能顾问-混沌演练（CFG）是一款遵循混沌工程实验原理并结合腾讯云内部实践的产品，提供基于真实线上故障的高可用能力演练服务，帮助用户的系统提升容错性和韧性。

- 本章节介绍的智能顾问-混沌演练 API 接口均为 API 3.0接口。
- 您可以调用 API 对智能顾问-混沌演练进行操作，例如查询混沌演练任务、创建混沌演练任务等。
- 智能顾问-混沌演练支持的所有接口信息，请参见 API 概览。

术语表

智能顾问-混沌演练 API 接口的常见术语请参见下表：

术语	描述
演练任务	演练任务包含演练基本信息、参与演练的实例、演练故障动作、可观测指标等。执行演练任务，即可启动混沌演练。
经验库	为了帮助用户快速创建演练，智能顾问-混沌演练提供了电商、游戏、多媒体等多个行业的演练经验模板，覆盖跨可用区容灾等多个典型应用场景。用户可以根据自身业务需求，快速高效地复用成熟解决方案，验证系统的可用性。
动作库	智能顾问-混沌演练提供包含计算、数据库、容器、网络、中间件等多个资源类型的故障动作库，可从故障动作库中选择指定动作创建演练任务。

使用限制

具体参数限制，请参考各接口文档中的参数说明。

API 快速入门

您可以使用 API Explorer 工具在线调用 API。
本文以查询演练任务为例，通过 API Explorer 工具调用 API 接口的步骤如下：

1. 进入 API Explorer 工具页面。更多 API Explorer 工具使用信息，请参见 使用 API Explorer。
2. 调用 DescribeTask 接口，填写公共参数，查询指定演练任务信息。
3. 在返回结果中查看演练任务详情。

API 概览

最近更新时间：2026-07-10 16:48:50

任务相关接口

接口名称	接口功能	频率限制（次/秒）
CreateTaskFromTemplate	从经验库创建演练	10
DeleteTask	删除任务	10
DescribeActionFieldConfigList	获取动作栏位配置参数列表	20
DescribeActionLibraryList	获取动作库列表	100
DescribeObjectTypeList	查询对象类型列表	20
DescribeTask	查询任务	20
DescribeTaskExecuteLogs	获取演练过程日志	20
DescribeTaskList	查询任务列表	100
DescribeTemplate	查询经验库	20
ExecuteTask	执行任务	10
ExecuteTaskInstance	执行任务动作实例	10
ModifyTaskRunStatus	修改任务运行状态	10
CreateTaskFromAction	从动作创建演练	20
DescribeTaskPolicyTriggerLog	获取护栏触发日志	20
TriggerPolicy	触发护栏策略	20

经验库相关接口

接口名称	接口功能	频率限制（次/秒）
DescribeTemplateList	查询经验库列表	100

调用方式
请求结构

最近更新时间：2026-07-10 16:48:50

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 cfg.intl.tencentcloudapi.com ，也支持指定地域域名访问，例如广州地域的域名为 cfg.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 cfg.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）	cfg.intl.tencentcloudapi.com
华南地区（广州）	cfg.ap-guangzhou.tencentcloudapi.com
华东地区（上海）	cfg.ap-shanghai.tencentcloudapi.com
华东地区（南京）	cfg.ap-nanjing.tencentcloudapi.com
华北地区（北京）	cfg.ap-beijing.tencentcloudapi.com
西南地区（成都）	cfg.ap-chengdu.tencentcloudapi.com
西南地区（重庆）	cfg.ap-chongqing.tencentcloudapi.com
港澳台地区（中国香港）	cfg.ap-hongkong.tencentcloudapi.com
亚太东南（新加坡）	cfg.ap-singapore.tencentcloudapi.com
亚太东南（雅加达）	cfg.ap-jakarta.tencentcloudapi.com
亚太东南（曼谷）	cfg.ap-bangkok.tencentcloudapi.com
亚太东北（首尔）	cfg.ap-seoul.tencentcloudapi.com

亚太东北（东京）	cfg.ap-tokyo.tencentcloudapi.com
美国东部（弗吉尼亚）	cfg.na-ashburn.tencentcloudapi.com
美国西部（硅谷）	cfg.na-siliconvalley.tencentcloudapi.com
南美地区（圣保罗）	cfg.sa-saopaulo.tencentcloudapi.com
欧洲地区（法兰克福）	cfg.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）	cfg.ap-shanghai-fsi.tencentcloudapi.com
华南地区（深圳金融）	cfg.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：2026-07-10 16:48:51

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。

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 cfg；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。
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********************************/20
18-10-09/cvm/tc3_request, SignedHeaders=content-type;host, Signature=5da7a33f6993
f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
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********************************/20
18-05-30/cvm/tc3_request, SignedHeaders=content-type;host, Signature=582c400e06b5
924a6f2b5d7d672d79c15b13162d9279b0855cfba6789a8edb4c
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********************************/20
18-05-30/cvm/tc3_request, SignedHeaders=content-type;host, Signature=582c400e06b5
924a6f2b5d7d672d79c15b13162d9279b0855cfba6789a8edb4c
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。

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。
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=37ac2f4fde00b0ac9bd9eadeb459b1bbee224158d66e7ae5fcadb70b2d18
1d02&Region=ap-guangzhou&Nonce=23823223&SecretId=AKID**************************
****
```

地域列表

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

地域	取值
华北地区（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
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

接口鉴权 v3

最近更新时间：2026-07-10 16:48:53

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 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 7 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++。

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, Signat
ure=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"}]}
```

⚠ Note：

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。当前支持的编程语言有：

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

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

- Signature Demo

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

Java

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

public class TencentCloudAPITC3Demo {
private final static Charset UTF8 = StandardCharsets.UTF_8;
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID**********************************
private final static String SECRET_ID = System.getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
private final static String SECRET_KEY = System.getenv("TENCENTCLOUD_SECRET_KEY");
private final static String CT_JSON = "application/json; charset=utf-8";
```

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

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

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

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

String payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u5
40d\"], \"Name\": \"instance-name\"}]}";
String hashedRequestPayload = sha256Hex(payload);
String canonicalRequest = httpRequestMethod + "\n" + canonicalUri + "\n" + canoni
calQueryString + "\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, string
ToSign)).toLowerCase();
System.out.println(signature);

// ************* 步骤 4: 拼接 Authorization *************
String authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + creden
tialScope + ", "
+ "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-nam
e"}]}

# ************* 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, act
ion.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")).hexd
igest()
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.sha2
56).hexdigest()
print(signature)

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

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

Golang

```
package main

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

func sha256hex(s string) string {
```

```
b := sha256.Sum256([]byte(s))
return hex.EncodeToString(b[:])
}

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

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

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

// step 2: build string to sign
date := time.Unix(timestamp, 0).UTC().Format("2006-01-02")
```

```
credentialScope := fmt.Sprintf("%s/%s/tc3_request", date, service)
hashedCanonicalRequest := sha256hex(canonicalRequest)
string2sign := fmt.Sprintf("%s\n%d\n%s\n%s",
algorithm,
timestamp,
credentialScope,
hashedCanonicalRequest)
fmt.Println(string2sign)

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

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

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

PHP

```
<?php
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID**********************
******
$secretId = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 **********************
***
$secretKey = getenv("TENCENTCLOUD_SECRET_KEY");
```

```
$host = "cvm.tencentcloudapi.com";
$service = "cvm";
$version = "2017-03-12";
$action = "DescribeInstances";
$region = "ap-guangzhou";
// $timestamp = time();
$timestamp = 1551113065;
$algorithm = "TC3-HMAC-SHA256";

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

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

// step 3: sign string
$secretDate = hash_hmac("SHA256", $date, "TC3".$secretKey, true);
```

```
$secretService = hash_hmac("SHA256", $service, $secretDate, true);
$secretSigning = hash_hmac("SHA256", "tc3_request", $secretService, true);
$signature = hash_hmac("SHA256", $stringToSign, $secretSigning);
echo $signature.PHP_EOL;

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

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

Ruby

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

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

service = 'cvm'
host = 'cvm.tencentcloudapi.com'
endpoint = 'https://' + host
region = 'ap-guangzhou'
action = 'DescribeInstances'
version = '2017-03-12'
algorithm = 'TC3-HMAC-SHA256'
```

```
# timestamp = Time.now.to_i
timestamp = 1551113065
date = Time.at(timestamp).utc.strftime('%Y-%m-%d')

# ************* 步骤 1：拼接规范请求串 *************
http_request_method = 'POST'
canonical_uri = '/'
canonical_querystring = ''
canonical_headers = "content-type:application/json; charset=utf-8\nhost:#{host}\n
x-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": "i
nstance-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}, Signed
Headers=#{signed_headers}, Signature=#{signature}"
puts authorization

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

DotNet

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

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

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

```
}

public static Dictionary<String, String> BuildHeaders(string secretid,
string secretkey, string service, string endpoint, string region,
string action, string version, DateTime date, string requestPayload)
{
string datestr = date.ToString("yyyy-MM-dd");
DateTime startTime = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc);
long requestTimestamp = (long)Math.Round((date - startTime).TotalMilliseconds, Mi
dpointRounding.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_requ
est"));
byte[] signatureBytes = HmacSHA256(secretSigning, Encoding.UTF8.GetBytes(stringTo
Sign));
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_KE
Y");

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\\u5
47d\\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 ×tamp)
{
string utcDate;
char buff[20] = {0};
// time_t timenow;
struct tm sttime;
sttime = *gmtime(×tamp);
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\\u5
40d\"], \"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" + credentialScop
e + "\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 + "/" + creden
tialScope + ", "
+ "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(×tamp);
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_le
n);
unsigned char k_signing[64] = {0};
hmac_sha256(k_service, output_len, "tc3_request", strlen("tc3_request"), k_signin
g, &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）。

接口鉴权

最近更新时间：2026-07-10 16:48:54

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 7 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++。

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=1465
185768&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&L
imit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKID******************
*************&Timestamp=1465185768&Version=2017-03-12
```

2.4. 生成签名串

此步骤生成签名串。

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

具体代码如下，以 PHP 语言为例：

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

最终得到的签名串为：

```
7RAM2xfNMO9EiVTNmPg06MRnCvQ=
```

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

3. 签名串编码

生成的签名串并不能直接作为请求参数，需要对其进行 URL 编码。

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

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

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

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

4. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理。

错误代码	错误描述
AuthFailure.SignatureExpire	签名过期
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。当前支持的编程语言有：

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

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.getAlg
orithm());
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 UnsupportedEnc
odingException {
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(), CHAR
SET)).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.currentTimeMil
lis() / 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**********************
******
```

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

ksort($param);

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

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

Ruby

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

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

```
**
secret_key = ENV["TENCENTCLOUD_SECRET_KEY"]

method = 'GET'
endpoint = 'cvm.tencentcloudapi.com'
data = {
'Action' => 'DescribeInstances',
'InstanceIds.0' => 'ins-09dx96dg',
'Limit' => 20,
'Nonce' => 11886,
'Offset' => 0,
'Region' => 'ap-guangzhou',
'SecretId' => secret_id,
'Timestamp' => 1465185768, # Time.now.to_i
'Version' => '2017-03-12',
}
sign = method + endpoint + '/?'
params = []
data.sort.each do |item|
params << "#{item[0]}=#{item[1]}"
end
sign += params.join('&')
digest = OpenSSL::Digest.new('sha1')
data['Signature'] = Base64.encode64(OpenSSL::HMAC.digest(digest, secret_key, sig
n))
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> requestPa
rams, string requestMethod, string requestHost, string requestPath)
{
string retStr = "";
retStr += requestMethod;
retStr += requestHost;
retStr += requestPath;
retStr += "?";
string v = "";
foreach (string key in requestParams.Keys)
{
v += string.Format("{0}={1}&", key, requestParams[key]);
}
retStr += v.TrimEnd('&');
return retStr;
}

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

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>(p
aram, 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：2026-07-10 16:48:54

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 si
gnature 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。

任务相关接口
从经验库创建演练

最近更新时间：2026-07-10 16:49:08

1. 接口描述

接口请求域名： cfg.intl.tencentcloudapi.com 。

从经验库创建演练

默认接口请求频率限制：10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：CreateTaskFromTemplate。
Version	是	String	公共参数，本接口取值：2021-08-20。
Region	否	String	公共参数，此参数为可选参数。
TemplateId	是	Integer	从经验库中查询到的经验模板ID
TaskConfig	是	TaskConfig	演练的配置参数

3. 输出参数

参数名称	类型	描述
TaskId	Integer	创建成功的演练ID

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 根据经验模板创建演练

通过预创建好的经验模板生成常规化的演练持续使用

输入示例

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

{
"TemplateId": 689,
"TaskConfig": {
"TaskTitle": "测试演练，关联了实例，修改了演练名称和第一个动作组中第一个动作的动作自定义参数",
"TaskGroupsConfig": [
{
"TaskGroupInstances": [
"ins-xxxxxxxx"
],
"TaskGroupActionsConfig": [
{
"TaskGroupActionOrder": 1,
"TaskGroupActionCustomConfiguration": "{\"timeout\":200,\"percentage\":80}"
}
]
}
]
}
}
```

输出示例

```
{
"Response": {
"RequestId": "f0aee8ac-2ed3-4a7f-a25b-f0d7d228dd30",
"TaskId": 50
```

```
}
}
```

5. 开发者资源

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
- Tencent Cloud SDK 3.0 for Java
- Tencent Cloud SDK 3.0 for PHP
- Tencent Cloud SDK 3.0 for Go
- Tencent Cloud SDK 3.0 for Node.js
- Tencent Cloud SDK 3.0 for .NET
- Tencent Cloud SDK 3.0 for C++

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
AuthFailure.UnauthorizedOperation	cam未授权操作。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
LimitExceeded	超过配额限制。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

删除任务

最近更新时间：2026-07-10 16:49:07

1. 接口描述

接口请求域名： cfg.intl.tencentcloudapi.com 。

删除任务

默认接口请求频率限制：10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：DeleteTask。
Version	是	String	公共参数，本接口取值：2021-08-20。
Region	否	String	公共参数，此参数为可选参数。
TaskId	是	Integer	任务ID

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 删除任务

输入示例

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

{
"TaskId": "222"
}
```

输出示例

```
{
"Response": {
"RequestId": "6549ed1a-911f-46dd-b6cd-2c02d5bd180f"
}
}
```

5. 开发者资源

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
- Tencent Cloud SDK 3.0 for Java
- Tencent Cloud SDK 3.0 for PHP
- Tencent Cloud SDK 3.0 for Go
- Tencent Cloud SDK 3.0 for Node.js
- Tencent Cloud SDK 3.0 for .NET
- Tencent Cloud SDK 3.0 for C++

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
FailedOperation	操作失败。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。

获取动作栏位配置参数列表

最近更新时间：2026-07-10 16:49:06

1. 接口描述

接口请求域名： cfg.intl.tencentcloudapi.com 。

根据动作ID获取动作栏位动态配置参数信息，里面包含动作自有和通用两部分参数。

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：DescribeActionFieldConfigList。
Version	是	String	公共参数，本接口取值：2021-08-20。
Region	否	String	公共参数，此参数为可选参数。
ActionIds.N	是	Array of Integer	动作ID列表
ObjectTypeId	是	Integer	对象类型ID

3. 输出参数

参数名称	类型	描述
Common	Array of ActionFieldConfigResult	通用栏位配置列表

Results	Array of ActionFieldConfigResult	动作栏位配置列表
ResourceOffline	Array of ResourceOffline	资源下线信息
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 示例

输入示例

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

{
"ObjectTypeId": "1",
"ActionIds": [
"1"
]
}
```

输出示例

```
{
"Response": {
"RequestId": "f3433a9a-e8fd-40b9-88e7-dd8b3f1a181f",
"Common": [
{
"ActionId": 1,
"ActionName": "关机",
"ConfigDetail": [
{
"Type": "number",
"Lable": "动作超时时间(s)",
"Field": "ActionTimeout",
"DefaultValue": "1800",
```

```
"Config": "{\"max\": 86400, \"min\": 0, \"tooltip\": \"动作的超时时间\"}",
"Required": 1,
"Validate": "{\"op\": \"==\", \"value\": \"指定个数重启\", \"relatedField\": \"reboot_mod\"}",
"Visible": "{\"op\": \"<\", \"type\": \"need_insert\", \"value\": 0, \"relatedField\": \"ActionTimeout\"}"
}
]
}
],
"Results": []
}
}
```


5. 开发者资源

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
- Tencent Cloud SDK 3.0 for Java
- Tencent Cloud SDK 3.0 for PHP
- Tencent Cloud SDK 3.0 for Go
- Tencent Cloud SDK 3.0 for Node.js
- Tencent Cloud SDK 3.0 for .NET
- Tencent Cloud SDK 3.0 for C++

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	参数取值错误。
MissingParameter	缺少参数错误。

获取动作库列表

最近更新时间：2026-07-10 16:49:06

1. 接口描述

接口请求域名： cfg.intl.tencentcloudapi.com 。

获取混沌演练平台的动作库列表

默认接口请求频率限制：100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：DescribeActionLibraryList。
Version	是	String	公共参数，本接口取值：2021-08-20。
Region	否	String	公共参数，此参数为可选参数。
Limit	是	Integer	0-100
Offset	是	Integer	默认值0
ObjectType	是	Integer	对象类型ID
Filters.N	否	Array of ActionFilter	Keyword取值{"动作名称": "a_title", "描述": "a_desc", "动作类型": "a_type", "创建时间": "a_create_time", "二级分类": "a_resource_type"}
Attribute.N	否	Array of Integer	动作分类，1表示故障动作，2表示恢复动作
ActionIds.N	否	Array of	筛选项 -动作ID

		Integer	

3. 输出参数

参数名称	类型	描述
Results	Array of ActionLibraryListResult	查询结果列表
Total	Integer	符合记录条数
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 动作库列表查询

动作库列表数据查询

输入示例

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

{
"Limit": 10,
"Offset": 0,
"Filters": [
{
"Keyword": "a_type",
"Values": [
"1"
]
}
],
"ObjectType": 1,
"Attribute": [
1
```

```
]
}
```

输出示例


```
{
"Response": {
"RequestId": "cZ6m0xYnnDreBNJm",
"Results": [
{
"ActionName": "关机（测试）",
"Desc": "对CVM进行关机操作",
"ActionType": "平台",
"CreateTime": "2023-07-04 11:20:43",
"Creator": "系统",
"UpdateTime": "2023-07-04 11:20:43",
"RiskDesc": "高风险",
"ActionId": 1,
"AttributeId": 1,
"RelationActionId": 2,
"ActionCommand": "调用腾讯云对应产品的API接口",
"ActionContent": "调用云api StopInstances",
"ActionCommandType": 1,
"ActionDetail": "<p>调用云api <a href=\"https://cloud.tencent.com/document/produc
t/213/15743\">StopInstances</a>关机</p>",
"ResourceType": "服务器资源",
"IsAllowed": true,
"ActionBestCase": "https://cloud.tencent.com/document/product/1500/74357",
"ObjectType": "CVM",
"MetricIdList": [
614,
615
]
},
{
"ActionName": "开机",
"Desc": "对CVM进行开机操作",
"ActionType": "平台",
"CreateTime": "2022-11-29 18:08:46",
"Creator": "系统",
"UpdateTime": "2022-11-29 18:08:46",
"RiskDesc": "高风险",
"ActionId": 2,
"AttributeId": 2,
"RelationActionId": 1,
"ActionCommand": "调用腾讯云对应产品的API接口",
```

```
"ActionContent": "调用云api StartInstances",
"ActionCommandType": 1,
"ActionDetail": "<p>调用云api <a href=\"https://cloud.tencent.com/document/product/213/15735\">StartInstances</a>开机</p>",
"ResourceType": "服务器资源",
"IsAllowed": true,
"ActionBestCase": "",
"ObjectType": "CVM",
"MetricIdList": []
},
{
"ActionName": "重启",
"Desc": "重启",
"ActionType": "平台",
"CreateTime": "2023-05-24 15:33:39",
"Creator": "系统",
"UpdateTime": "2023-05-24 15:33:39",
"RiskDesc": "高风险",
"ActionId": 3,
"AttributeId": 1,
"RelationActionId": 0,
"ActionCommand": "调用腾讯云对应产品的API接口",
"ActionContent": "调用云api RebootInstances",
"ActionCommandType": 1,
"ActionDetail": "<p>调用云api <a href=\"https://cloud.tencent.com/document/product/213/15742\">RebootInstances</a>重启</p>",
"ResourceType": "服务器资源",
"IsAllowed": true,
"ActionBestCase": "https://tcloud4api.woa.com/document/product/1607/88863?!preview&!document=1",
"ObjectType": "CVM",
"MetricIdList": []
},
{
"ActionName": "CPU利用率高",
"Desc": "使用stress-ng压测工具压测，支持的linux发行版：Centos7.2及以上，CoreOS 1745.5.0及以上，Debian9.0及以上，Ubuntu 16.04.1及以上",
"ActionType": "平台",
"CreateTime": "2022-11-29 18:08:46",
"Creator": "系统",
"UpdateTime": "2022-11-29 18:08:46",
"RiskDesc": "高风险",
"ActionId": 4,
"AttributeId": 1,
"RelationActionId": 0,
"ActionCommand": "#!/bin/bash\n\nuser=$(whoami)\nif [ !$user == 'root' ]\nthen\n sudo -i\nfi\n\nfunction command_exists(){\n if command -v $1 > /dev/null 2>&1; th
```

```
en\n return 1\n else\n return 0\n fi\n}\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n\nos_desc=$(cat /etc/*release)\n# echo $os_desc\n\ndeclare -A os_dic\nos_dic=([CentOS]=\"yum install stress-ng -y\" \\\n [CoreOS]=\"docker pull alexeiled/stress-ng\" \\\n [Debian]=\"apt-get install -y stress-ng\" \\\n [Ubuntu]=\"apt-get install -y stress-ng\")\n\nos_name='N/A'\n\nfor key in $(echo ${!os_dic[*]})\ndo\n if [[ $os_desc =~ $key ]]\n then\n os_name=$key\n echo $key\n fi\ndone\n\nif [ $os_name == 'N/A' ]\nthen\n echo \"Unsupported Linux distributions\"\n exit 1\nfi\n\n# os_name=$(cat /etc/*release | awk 'NR==1{print($1)}')\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"installing stress-ng...\"\ncommand_exists stress-ng\n\nif [[ $? -eq 0 ]]\nthen\n ${os_dic[$os_name]}\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Checking the installation status...\"\n command_exists stress-ng\n if [[ $? -eq 0 && $os_name != 'CoreOS' ]]\n then\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Fail to install\"\n exit 1\n else\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"stress-ng has already installed\"\n fi\nelse\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"stress-ng has already installed\"\nfi\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"Starting to perform stress test.\"\nif [ $os_name == \"CoreOS\" ]\nthen\n docker run --rm alexeiled/stress-ng -c 0 -l {{percentage}} --timeout {{timeout}}\nelse\n stress-ng -c 0 -l {{percentage}} --timeout {{timeout}}\nfi\n\nif [[ !$? -eq 0 ]]\nthen\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"failed\"\n exit 1\nelse\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"completed\"\n exit 0\nfi",
"ActionContent": "#!/bin/bash\n\nuser=$(whoami)\nif [ !$user == 'root' ]\nthen\n sudo -i\nfi\n\nfunction command_exists(){\n if command -v $1 > /dev/null 2>&1; then\n return 1\n else\n return 0\n fi\n}\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n\nos_desc=$(cat /etc/*release)\n# echo $os_desc\n\ndeclare -A os_dic\nos_dic=([CentOS]=\"yum install stress-ng -y\" \\\n [CoreOS]=\"docker pull alexeiled/stress-ng\" \\\n [Debian]=\"apt-get install -y stress-ng\" \\\n [Ubuntu]=\"apt-get install -y stress-ng\")\n\nos_name='N/A'\n\nfor key in $(echo ${!os_dic[*]})\ndo\n if [[ $os_desc =~ $key ]]\n then\n os_name=$key\n echo $key\n fi\ndone\n\nif [ $os_name == 'N/A' ]\nthen\n echo \"Unsupported Linux distributions\"\n exit 1\nfi\n\n# os_name=$(cat /etc/*release | awk 'NR==1{print($1)}')\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"installing stress-ng...\"\ncommand_exists stress-ng\n\nif [[ $? -eq 0 ]]\nthen\n ${os_dic[$os_name]}\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Checking the installation status...\"\n command_exists stress-ng\n if [[ $? -eq 0 && $os_name != 'CoreOS' ]]\n then\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Fail to install\"\n exit 1\n else\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"stress-ng has already installed\"\n fi\nelse\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"stress-ng has already installed\"\nfi\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"Starting to perform stress test.\"\nif [ $os_name == \"CoreOS\" ]\nthen\n docker run --rm alexeiled/stress-ng -c 0 -l {{percentage}} --timeout {{timeout}}\nelse\n stress-ng -c 0 -l {{percentage}} --timeout {{timeout}}\nfi\n\nif [[ !$? -eq 0 ]]\nthen\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"failed\"\n exit 1\nelse\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"completed\"\n exit 0\nfi",
"ActionCommandType": 0,
```

```
"ActionDetail": "<p>使用tat通道下发stress-ng命令进行压测</p>\n<p>命令内容：stress-ng -c 0 -l {{percentage}} --timeout {{timeout}}</p>\n<p><a href=\"https://cloud.tencent.com/document/api/1340/52676\">tat官方文档</a></p>\n<p><a href=\"https://wiki.ubuntu.com/Kernel/Reference/stress-ng\">stress-ng官方文档</a></p>\n",
"ResourceType": "CPU资源",
"IsAllowed": true,
"ActionBestCase": "",
"ObjectType": "CVM",
"MetricIdList": []
},
{
"ActionName": "内存利用率高",
"Desc": "使用stress-ng压测工具压测，支持的linux发行版：Centos7.2及以上，CoreOS 1745.5.0及以上，Debian9.0及以上，Ubuntu 16.04.1及以上",
"ActionType": "平台",
"CreateTime": "2022-11-29 18:08:46",
"Creator": "系统",
"UpdateTime": "2022-11-29 18:08:46",
"RiskDesc": "高风险",
"ActionId": 7,
"AttributeId": 1,
"RelationActionId": 0,
"ActionCommand": "#!/bin/bash\n\nuser=$(whoami)\nif [ !$user == 'root' ]\nthen\n sudo -i\nfi\n\nfunction command_exists(){\n if command -v $1 > /dev/null 2>&1; then\n return 1\n else\n return 0\n fi\n}\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n\nos_desc=$(cat /etc/*release)\n# echo $os_desc\n\ndeclare -A os_dic\nos_dic=([CentOS]=\"yum install stress-ng -y\" \\\n [CoreOS]=\"docker pull alexeiled/stress-ng\" \\\n [Debian]=\"apt-get install -y stress-ng\" \\\n [Ubuntu]=\"apt-get install -y stress-ng\")\n\nos_name='N/A'\n\nfor key in $(echo ${!os_dic[*]})\ndo\n if [[ $os_desc =~ $key ]]\n then\n os_name=$key\n echo $key\n fi\ndone\n\nif [ $os_name == 'N/A' ]\nthen\n echo \"Unsupported Linux distributions\"\n exit 1\nfi\n\n# os_name=$(cat /etc/*release | awk 'NR==1{print($1)}')\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"installing stress-ng...\"\ncommand_exists stress-ng\n\nif [[ $? -eq 0 ]]\nthen\n ${os_dic[$os_name]}\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Checking the installation status...\"\n command_exists stress-ng\n if [[ $? -eq 0 && $os_name != 'CoreOS' ]]\n then\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Fail to install\"\n exit 1\n else\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"stress-ng has already installed\"\n fi\nelse\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"stress-ng has already installed\"\nfi\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"Starting to perform stress test\"\nif [ $os_name == \"CoreOS\" ]\nthen\n docker run --rm alexeiled/stress-ng --vm-bytes $(awk '/MemAvailable/{printf \"%d\\n\", $2 * 0.01*{{percentage}};}' < /proc/meminfo)k --vm-keep -m 1 -t {{timeout}}\nelse\n avaliable_mem=$(awk '/^MemAvailable:/{printf \"%d\\n\", $2;}' < /proc/meminfo)\n free_mem=$(awk '/^MemFree:/{printf \"%d\\n\", $2;}' < /proc/meminfo)\n buffer=$(awk '/^Buffers:/{printf \"%d\\n\", $2;}' < /proc/meminfo)\n cache=$(awk '/^Cached:/{printf \"%d\\n\", $2;}' <
```

```
/proc/meminfo)\n if [ -n \"$avaliable_mem\" ]\n then\n stress-ng --vm-bytes $[$av
aliable_mem*{{percentage}}/100]k --vm-keep -m 1 -t {{timeout}}\n else\n stress-ng
--vm-bytes $[($free_mem+$buffer+$cache)*{{percentage}}/100]k --vm-keep -m 1 -t
{{timeout}}\n fi\nfi\n\nif [[ !$? -eq 0 ]]\nthen\n echo -e \"[\"`date +\"%Y-%m-%d
%H:%M:%S\"`\"] \\c\"\n echo \"Failed\"\n exit 1\nelse\n echo -e \"[\"`date +\"%Y-
%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Completed\"\n exit 0\nfi",
"ActionContent": "#!/bin/bash\n\nuser=$(whoami)\nif [ !$user == 'root' ]\nthen\n
sudo -i\nfi\n\nfunction command_exists(){\n if command -v $1 > /dev/null 2>&1; th
en\n return 1\n else\n return 0\n fi\n}\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%
S\"`\"] \\c\"\n\nos_desc=$(cat /etc/*release)\n# echo $os_desc\n\ndeclare -A os_d
ic\nos_dic=([CentOS]=\"yum install stress-ng -y\" \\\n [CoreOS]=\"docker pull ale
xeiled/stress-ng\" \\\n [Debian]=\"apt-get install -y stress-ng\" \\\n [Ubuntu]=
\"apt-get install -y stress-ng\")\n\nos_name='N/A'\n\nfor key in $(echo ${!os_dic
[*]})\ndo\n if [[ $os_desc =~ $key ]]\n then\n os_name=$key\n echo $key\n fi\ndon
e\n\nif [ $os_name == 'N/A' ]\nthen\n echo \"Unsupported Linux distributions\"\n
exit 1\nfi\n\n# os_name=$(cat /etc/*release | awk 'NR==1{print($1)}')\n\necho -e
\"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"installing stress-ng...\"\nco
mmand_exists stress-ng\n\nif [[ $? -eq 0 ]]\nthen\n ${os_dic[$os_name]}\n echo -e
\"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Checking the installation st
atus...\"\n command_exists stress-ng\n if [[ $? -eq 0 && $os_name != 'CoreOS' ]]
\n then\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Fail to in
stall\"\n exit 1\n else\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n e
cho \"stress-ng has already installed\"\n fi\nelse\n echo -e \"[\"`date +\"%Y-%m-
%d %H:%M:%S\"`\"] \\c\"\n echo \"stress-ng has already installed\"\nfi\n\necho -e
\"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"Starting to perform stress te
st\"\nif [ $os_name == \"CoreOS\" ]\nthen\n docker run --rm alexeiled/stress-ng -
-vm-bytes $(awk '/MemAvailable/{printf \"%d\\n\", $2 * 0.01*{{percentage}};}' < /
proc/meminfo)k --vm-keep -m 1 -t {{timeout}}\nelse\n avaliable_mem=$(awk '/^MemAv
ailable:/{printf \"%d\\n\", $2;}' < /proc/meminfo)\n free_mem=$(awk '/^MemFree:/
{printf \"%d\\n\", $2;}' < /proc/meminfo)\n buffer=$(awk '/^Buffers:/{printf \"%d
\\n\", $2;}' < /proc/meminfo)\n cache=$(awk '/^Cached:/{printf \"%d\\n\", $2;}' <
/proc/meminfo)\n if [ -n \"$avaliable_mem\" ]\n then\n stress-ng --vm-bytes $[$av
aliable_mem*{{percentage}}/100]k --vm-keep -m 1 -t {{timeout}}\n else\n stress-ng
--vm-bytes $[($free_mem+$buffer+$cache)*{{percentage}}/100]k --vm-keep -m 1 -t
{{timeout}}\n fi\nfi\n\nif [[ !$? -eq 0 ]]\nthen\n echo -e \"[\"`date +\"%Y-%m-%d
%H:%M:%S\"`\"] \\c\"\n echo \"Failed\"\n exit 1\nelse\n echo -e \"[\"`date +\"%Y-
%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Completed\"\n exit 0\nfi",
"ActionCommandType": 0,
"ActionDetail": "<p>使用tat通道下发stress-ng命令进行压测</p>\n<p>命令内容：stress-ng -
-vm-bytes $(awk '/MemAvailable/{printf \"%d\\n\", $2 * 0.01*{{percentage}};}' < /
proc/meminfo)k --vm-keep -m 1 -t {{timeout}}</p>\n<p><a href=\"https://cloud.tenc
ent.com/document/api/1340/52676\">tat官方文档</a></p>\n<p><a href=\"https://wiki.u
buntu.com/Kernel/Reference/stress-ng\">stress-ng官方文档</a></p>",
"ResourceType": "内存资源",
"IsAllowed": true,
"ActionBestCase": "",
"ObjectType": "CVM",
```

```
"MetricIdList": []
},
{
"ActionName": "磁盘分区使用率高",
"Desc": "使用stress-ng压测工具压测，支持的linux发行版：Centos7.2及以上，CoreOS 1745.5.0及以上，Debian9.0及以上，Ubuntu 16.04.1及以上",
"ActionType": "平台",
"CreateTime": "2022-11-29 18:08:46",
"Creator": "系统",
"UpdateTime": "2022-11-29 18:08:46",
"RiskDesc": "高风险",
"ActionId": 8,
"AttributeId": 1,
"RelationActionId": 0,
"ActionCommand": "#!/bin/bash\n\nuser=$(whoami)\nif [ !$user == 'root' ]\nthen\n sudo -i\nfi\n\nfunction command_exists(){\n if command -v $1 > /dev/null 2>&1; then\n return 1\n else\n return 0\n fi\n}\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n\nos_desc=$(cat /etc/*release)\n# echo $os_desc\n\ndeclare -A os_dic\nos_dic=([CentOS]=\"yum install stress-ng -y\" \\\n [CoreOS]=\"docker pull alexeiled/stress-ng\" \\\n [Debian]=\"apt-get install -y stress-ng\" \\\n [Ubuntu]=\"apt-get install -y stress-ng\")\n\nos_name='N/A'\n\nfor key in $(echo ${!os_dic[*]})\ndo\n if [[ $os_desc =~ $key ]]\n then\n os_name=$key\n echo $key\n fi\ndone\n\nif [ $os_name == 'N/A' ]\nthen\n echo \"Unsupported Linux distributions\"\n exit 1\nfi\n\n# os_name=$(cat /etc/*release | awk 'NR==1{print($1)}')\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"installing stress-ng...\"\ncommand_exists stress-ng\n\nif [[ $? -eq 0 ]]\nthen\n ${os_dic[$os_name]}\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Checking the installation status...\"\n command_exists stress-ng\n if [[ $? -eq 0 && $os_name != 'CoreOS' ]]\n then\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Fail to install\"\n exit 1\n else\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"stress-ng has already installed\"\n fi\nelse\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"stress-ng has already installed\"\nfi\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"Starting to perform stress test.\"\nif [ $os_name == \"CoreOS\" ]\nthen\n cd {{dir}} && docker run --rm alexeiled/stress-ng --iomix 1 --iomix-bytes `df -k {{dir}} | awk 'NR==2{printf(\"%d\\n\", ($3+$4)*0.01*{{percentage}}-$3)}'`k -t {{timeout}}\nelse\n cd {{dir}} && stress-ng --iomix 1 --iomix-bytes `df -k {{dir}} | awk 'NR==2{printf(\"%d\\n\", ($3+$4)*0.01*{{percentage}}-$3)}'`k -t {{timeout}}\nfi\n\nif [[ !$? -eq 0 ]]\nthen\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Failed\"\n exit 1\nelse\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Completed\"\n exit 0\nfi",
"ActionContent": "#!/bin/bash\n\nuser=$(whoami)\nif [ !$user == 'root' ]\nthen\n sudo -i\nfi\n\nfunction command_exists(){\n if command -v $1 > /dev/null 2>&1; then\n return 1\n else\n return 0\n fi\n}\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n\nos_desc=$(cat /etc/*release)\n# echo $os_desc\n\ndeclare -A os_dic\nos_dic=([CentOS]=\"yum install stress-ng -y\" \\\n [CoreOS]=\"docker pull alexeiled/stress-ng\" \\\n [Debian]=\"apt-get install -y stress-ng\" \\\n [Ubuntu]=
```

\"apt-get install -y stress-ng\")\n\nos_name='N/A'\n\nfor key in $(echo ${!os_dic[*]})\ndo\n if [[$os_desc =~ $key]]\n then\n os_name=$key\n echo $key\n fi\ndone\n\nif [$os_name == 'N/A']\nthen\n echo \"Unsupported Linux distributions\"\n exit 1\nfi\n\n# os_name=$(cat /etc/*release | awk 'NR==1{print($1)}')\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"installing stress-ng...\"\ncommand_exists stress-ng\n\nif [[$? -eq 0]]\nthen\n ${os_dic[$os_name]}\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Checking the installation status...\"\n command_exists stress-ng\n if [[$? -eq 0 && $os_name != 'CoreOS']]\n then\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Fail to install\"\n exit 1\n else\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"stress-ng has already installed\"\n fi\nelse\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"stress-ng has already installed\"\nfi\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"Starting to perform stress test.\"\nif [$os_name == \"CoreOS\"]\nthen\n cd {{dir}} && docker run --rm alexeiled/stress-ng --iomix 1 --iomix-bytes `df -k {{dir}} | awk 'NR==2{printf(\"%d\\n\", ($3+$4)*0.01*{{percentage}}-$3)}'`k -t {{timeout}}\nelse\n cd {{dir}} && stress-ng --iomix 1 --iomix-bytes `df -k {{dir}} | awk 'NR==2{printf(\"%d\\n\", ($3+$4)*0.01*{{percentage}}-$3)}'`k -t {{timeout}}\nfi\n\nif [[!$? -eq 0]]\nthen\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Failed\"\n exit 1\nelse\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Completed\"\n exit 0\nfi",
"ActionCommandType": 0,
"ActionDetail": " 使用tat通道下发stress-ng命令进行压测 \n 命令内容：stress-ng --iomix 1 --iomix-bytes `df -k {{dir}} | awk 'NR==2{printf(\"%d\\n\", ($3+$4)*0.01*{{percentage}}-$3)}'`k -t {{timeout}} \n tat官方文档 \n stress-ng官方文档 \n",
"ResourceType": "磁盘资源",
"IsAllowed": true,
"ActionBestCase": "",
"ObjectType": "CVM",
"MetricIdList": []
},
{
"ActionName": "内核故障",
"Desc": "会触发实例重启",
"ActionType": "平台",
"CreateTime": "2022-11-29 18:08:46",
"Creator": "系统",
"UpdateTime": "2022-11-29 18:08:46",
"RiskDesc": "高风险",
"ActionId": 9,
"AttributeId": 1,
"RelationActionId": 24,
"ActionCommand": "echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\" && echo 'inject kernel error success!' && echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\" && sleep 5 && echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\" && echo c >

```
/proc/sysrq-trigger &",
"ActionContent": "echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\" && echo 'in
ject kernel error success!' && echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c
\" && sleep 5 && echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\" && echo c >
/proc/sysrq-trigger &",
"ActionCommandType": 0,
"ActionDetail": "<p>使用tat通道下发内核故障命令</p>\n<p>命令内容：echo c > /proc/sysrq
-trigger</p>\n<p><a href=\"https://cloud.tencent.com/document/api/1340/52676\">ta
t官方文档</a></p>",
"ResourceType": "CPU资源",
"IsAllowed": true,
"ActionBestCase": "",
"ObjectType": "CVM",
"MetricIdList": []
},
{
"ActionName": "磁盘IO负载",
"Desc": "使用stress-ng压测工具压测，支持的linux发行版：Centos7.2及以上，CoreOS 1745.5.0
及以上，Debian9.0及以上，Ubuntu 16.04.1及以上",
"ActionType": "平台",
"CreateTime": "2022-11-29 18:08:46",
"Creator": "系统",
"UpdateTime": "2022-11-29 18:08:46",
"RiskDesc": "高风险",
"ActionId": 10,
"AttributeId": 1,
"RelationActionId": 0,
"ActionCommand": "#!/bin/bash\n\nuser=$(whoami)\nif [ !$user == 'root' ]\nthen\n
sudo -i\nfi\n\nfunction command_exists(){\n if command -v $1 > /dev/null 2>&1; th
en\n return 1\n else\n return 0\n fi\n}\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%
S\"`\"] \\c\"\n\nos_desc=$(cat /etc/*release)\n# echo $os_desc\n\ndeclare -A os_d
ic\nos_dic=([CentOS]=\"yum install stress-ng -y\" \\\n [CoreOS]=\"docker pull ale
xeiled/stress-ng\" \\\n [Debian]=\"apt-get install -y stress-ng\" \\\n [Ubuntu]=
\"apt-get install -y stress-ng\")\n\nos_name='N/A'\n\nfor key in $(echo ${!os_dic
[*]})\ndo\n if [[ $os_desc =~ $key ]]\n then\n os_name=$key\n echo $key\n fi\ndon
e\n\nif [ $os_name == 'N/A' ]\nthen\n echo \"Unsupported Linux distributions\"\n
exit 1\nfi\n\n# os_name=$(cat /etc/*release | awk 'NR==1{print($1)}')\n\necho -e
\"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"installing stress-ng...\"\nco
mmand_exists stress-ng\n\nif [[ $? -eq 0 ]]\nthen\n ${os_dic[$os_name]}\n echo -e
\"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Checking the installation st
atus...\"\n command_exists stress-ng\n if [[ $? -eq 0 && $os_name != 'CoreOS' ]]
\n then\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Fail to in
stall\"\n exit 1\n else\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n e
cho \"stress-ng has already installed\"\n fi\nelse\n echo -e \"[\"`date +\"%Y-%m-
%d %H:%M:%S\"`\"] \\c\"\n echo \"stress-ng has already installed\"\nfi\n\necho -e
\"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"Starting to perform stress te
st.\"\nif [ $os_name == \"CoreOS\" ]\nthen\n docker run --rm alexeiled/stress-ng
```

```
--vm-bytes {{bytes_num}}{{bytes_unit}} --hdd {{io_process_num}} --vm-keep -t {{ti
meout}}\nelse\n stress-ng --vm-bytes {{bytes_num}}{{bytes_unit}} --hdd {{io_proce
ss_num}} --vm-keep -t {{timeout}}\nfi\n\nif [[ !$? -eq 0 ]]\nthen\n echo -e \"
[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Failed\"\n exit 1\nelse\n echo
-e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Completed\"\n exit 0\nf
i",
"ActionContent": "#!/bin/bash\n\nuser=$(whoami)\nif [ !$user == 'root' ]\nthen\n
sudo -i\nfi\n\nfunction command_exists(){\n if command -v $1 > /dev/null 2>&1; th
en\n return 1\n else\n return 0\n fi\n}\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%
S\"`\"] \\c\"\n\nos_desc=$(cat /etc/*release)\n# echo $os_desc\n\ndeclare -A os_d
ic\nos_dic=([CentOS]=\"yum install stress-ng -y\" \\\n [CoreOS]=\"docker pull ale
xeiled/stress-ng\" \\\n [Debian]=\"apt-get install -y stress-ng\" \\\n [Ubuntu]=
\"apt-get install -y stress-ng\")\n\nos_name='N/A'\n\nfor key in $(echo ${!os_dic
[*]})\ndo\n if [[ $os_desc =~ $key ]]\n then\n os_name=$key\n echo $key\n fi\ndon
e\n\nif [ $os_name == 'N/A' ]\nthen\n echo \"Unsupported Linux distributions\"\n
exit 1\nfi\n\n# os_name=$(cat /etc/*release | awk 'NR==1{print($1)}')\n\necho -e
\"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"installing stress-ng...\"\nco
mmand_exists stress-ng\n\nif [[ $? -eq 0 ]]\nthen\n ${os_dic[$os_name]}\n echo -e
\"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Checking the installation st
atus...\"\n command_exists stress-ng\n if [[ $? -eq 0 && $os_name != 'CoreOS' ]]
\n then\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Fail to in
stall\"\n exit 1\n else\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n e
cho \"stress-ng has already installed\"\n fi\nelse\n echo -e \"[\"`date +\"%Y-%m-
%d %H:%M:%S\"`\"] \\c\"\n echo \"stress-ng has already installed\"\nfi\n\necho -e
\"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"Starting to perform stress te
st.\"\nif [ $os_name == \"CoreOS\" ]\nthen\n docker run --rm alexeiled/stress-ng
--vm-bytes {{bytes_num}}{{bytes_unit}} --hdd {{io_process_num}} --vm-keep -t {{ti
meout}}\nelse\n stress-ng --vm-bytes {{bytes_num}}{{bytes_unit}} --hdd {{io_proce
ss_num}} --vm-keep -t {{timeout}}\nfi\n\nif [[ !$? -eq 0 ]]\nthen\n echo -e \"
[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Failed\"\n exit 1\nelse\n echo
-e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Completed\"\n exit 0\nf
i",
"ActionCommandType": 0,
"ActionDetail": "<p>使用tat通道下发stress-ng命令进行压测</p>\n<p>命令内容：stress-ng -
-vm-bytes {{bytes_num}}{{bytes_unit}} --hdd {{io_process_num}} --vm-keep -t {{tim
eout}}</p>\n<p><a href=\"https://cloud.tencent.com/document/api/1340/52676\">tat
官方文档</a></p>",
"ResourceType": "IO资源",
"IsAllowed": true,
"ActionBestCase": "",
"ObjectType": "CVM",
"MetricIdList": []
},
{
"ActionName": "内存OOM",
"Desc": "使用stress-ng压测工具压测，支持的linux发行版：Centos7.2及以上，CoreOS 1745.5.0
及以上，Debian9.0及以上，Ubuntu 16.04.1及以上",
```

```
"ActionType": "平台",
"CreateTime": "2023-07-04 11:20:43",
"Creator": "系统",
"UpdateTime": "2023-07-04 11:20:43",
"RiskDesc": "高风险",
"ActionId": 11,
"AttributeId": 1,
"RelationActionId": 0,
"ActionCommand": "#!/bin/bash\n\nuser=$(whoami)\nif [ !$user == 'root' ]\nthen\n
sudo -i\nfi\n\nfunction command_exists(){\n if command -v $1 > /dev/null 2>&1; th
en\n return 1\n else\n return 0\n fi\n}\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%
S\"`\"] \\c\"\n\nos_desc=$(cat /etc/*release)\n# echo $os_desc\n\ndeclare -A os_d
ic\nos_dic=([CentOS]=\"yum install stress-ng -y\" \\\n [CoreOS]=\"docker pull ale
xeiled/stress-ng\" \\\n [Debian]=\"apt-get install -y stress-ng\" \\\n [Ubuntu]=
\"apt-get install -y stress-ng\")\n\nos_name='N/A'\n\nfor key in $(echo ${!os_dic
[*]})\ndo\n if [[ $os_desc =~ $key ]]\n then\n os_name=$key\n echo $key\n fi\ndon
e\n\nif [ $os_name == 'N/A' ]\nthen\n echo \"Unsupported Linux distributions\"\n
exit 1\nfi\n\n# os_name=$(cat /etc/*release | awk 'NR==1{print($1)}')\n\necho -e
\"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"installing stress-ng...\"\nco
mmand_exists stress-ng\n\nif [[ $? -eq 0 ]]\nthen\n ${os_dic[$os_name]}\n echo -e
\"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Checking the installation st
atus...\"\n command_exists stress-ng\n if [[ $? -eq 0 && $os_name != 'CoreOS' ]]
\n then\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Fail to in
stall\"\n exit 1\n else\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n e
cho \"stress-ng has already installed\"\n fi\nelse\n echo -e \"[\"`date +\"%Y-%m-
%d %H:%M:%S\"`\"] \\c\"\n echo \"stress-ng has already installed\"\nfi\n\necho -e
\"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"Starting to perform stress te
st.\"\nif [ $os_name == \"CoreOS\" ]\nthen\n docker run --rm alexeiled/stress-ng
--bigheap 10 --bigheap-growth 4K -t {{timeout}}\nelse\n stress-ng --bigheap 10 --
bigheap-growth 4K -t {{timeout}}\nfi\necho \"Completed\"\n\n",
"ActionContent": "#!/bin/bash\n\nuser=$(whoami)\nif [ !$user == 'root' ]\nthen\n
sudo -i\nfi\n\nfunction command_exists(){\n if command -v $1 > /dev/null 2>&1; th
en\n return 1\n else\n return 0\n fi\n}\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%
S\"`\"] \\c\"\n\nos_desc=$(cat /etc/*release)\n# echo $os_desc\n\ndeclare -A os_d
ic\nos_dic=([CentOS]=\"yum install stress-ng -y\" \\\n [CoreOS]=\"docker pull ale
xeiled/stress-ng\" \\\n [Debian]=\"apt-get install -y stress-ng\" \\\n [Ubuntu]=
\"apt-get install -y stress-ng\")\n\nos_name='N/A'\n\nfor key in $(echo ${!os_dic
[*]})\ndo\n if [[ $os_desc =~ $key ]]\n then\n os_name=$key\n echo $key\n fi\ndon
e\n\nif [ $os_name == 'N/A' ]\nthen\n echo \"Unsupported Linux distributions\"\n
exit 1\nfi\n\n# os_name=$(cat /etc/*release | awk 'NR==1{print($1)}')\n\necho -e
\"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"installing stress-ng...\"\nco
mmand_exists stress-ng\n\nif [[ $? -eq 0 ]]\nthen\n ${os_dic[$os_name]}\n echo -e
\"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Checking the installation st
atus...\"\n command_exists stress-ng\n if [[ $? -eq 0 && $os_name != 'CoreOS' ]]
\n then\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n echo \"Fail to in
stall\"\n exit 1\n else\n echo -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\n e
cho \"stress-ng has already installed\"\n fi\nelse\n echo -e \"[\"`date +\"%Y-%m-
```

```
%d %H:%M:%S\"`\"] \\c\"\n echo \"stress-ng has already installed\"\nfi\n\necho -e \"[\"`date +\"%Y-%m-%d %H:%M:%S\"`\"] \\c\"\necho \"Starting to perform stress test.\"\nif [ $os_name == \"CoreOS\" ]\nthen\n docker run --rm alexeiled/stress-ng --bigheap 10 --bigheap-growth 4K -t {{timeout}}\nelse\n stress-ng --bigheap 10 --bigheap-growth 4K -t {{timeout}}\nfi\necho \"Completed\"\n\n",
"ActionCommandType": 0,
"ActionDetail": "<p>使用tat通道下发stress-ng命令进行压测</p>\n<p>命令内容：stress-ng --bigheap 10 --bigheap-growth 4K -t {{timeout}}</p>\n<p><a href=\"https://cloud.tencent.com/document/api/1340/52676\">tat官方文档</a></p>",
"ResourceType": "内存资源",
"IsAllowed": true,
"ActionBestCase": "https://cloud.tencent.com/document/product/1500/81504",
"ObjectType": "CVM",
"MetricIdList": []
},
{
"ActionName": "空操作",
"Desc": "空操作，用于测试流程，不做实际注入操作",
"ActionType": "平台",
"CreateTime": "2022-11-29 18:08:46",
"Creator": "系统",
"UpdateTime": "2022-11-29 18:08:46",
"RiskDesc": "低风险",
"ActionId": 12,
"AttributeId": 1,
"RelationActionId": 13,
"ActionCommand": "调用腾讯云对应产品的API接口",
"ActionContent": "空操作",
"ActionCommandType": 1,
"ActionDetail": null,
"ResourceType": "其他",
"IsAllowed": true,
"ActionBestCase": "",
"ObjectType": "CVM",
"MetricIdList": []
}
],
"Total": 45
}
}
```


5. 开发者资源

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
- Tencent Cloud SDK 3.0 for Java
- Tencent Cloud SDK 3.0 for PHP
- Tencent Cloud SDK 3.0 for Go
- Tencent Cloud SDK 3.0 for Node.js
- Tencent Cloud SDK 3.0 for .NET
- Tencent Cloud SDK 3.0 for C++

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
MissingParameter	缺少参数错误。
UnsupportedOperation	操作不支持。

查询对象类型列表

最近更新时间：2026-07-10 16:49:05

1. 接口描述

接口请求域名： cfg.intl.tencentcloudapi.com 。

查询对象类型列表

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeObjectTypeList。
Version	是	String	公共参数，本接口取值：2021-08-20。
Region	否	String	公共参数，此参数为可选参数。
SupportType	否	Integer	所支持的对象 0：全平台产品 1：平台接入的对象 2：应用所支持的部分对象

3. 输出参数

参数名称	类型	描述
ObjectTypeList	Array of ObjectType	对象类型列表

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 请求对象类型列表

输入示例

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

{
"SupportType": 1
}
```

输出示例

```
{
"Response": {
"ObjectTypeList": [
{
"ArchLayer": 2,
"ObjectHasNewAction": false,
"ObjectPlatformName": "CVM",
"ObjectSupportType": 2,
"ObjectTypeId": 1,
"ObjectTypeJsonParse": null,
"ObjectTypeLevelOne": "计算",
"ObjectTypeParams": {
"Fields": [
{
"Header": "实例ID",
"JsonParse": null,
"Key": "InstanceId",
"Transfer": null,
"Type": 0
},
{
```

```
"Header": "实例名称",
"JsonParse": null,
"Key": "InstanceName",
"Transfer": null,
"Type": 0
},
{
"Header": "可用区",
"JsonParse": null,
"Key": "Placement.Zone",
"Transfer": "{\"ap-seoul-1\": \"\\u9996\\u5c14\\u4e00\\u533a\", \"ap-seoul-2\":
\"\\u9996\\u5c14\\u4e8c\\u533a\"}",
"Type": 0
},
{
"Header": "主IPv4地址",
"JsonParse": null,
"Key": "PrivateIpAddresses",
"Transfer": null,
"Type": 0
},
{
"Header": "VPC-ID",
"JsonParse": null,
"Key": "VirtualPrivateCloud.VpcId",
"Transfer": null,
"Type": 0
},
{
"Header": "子网ID",
"JsonParse": null,
"Key": "VirtualPrivateCloud.SubnetId",
"Transfer": null,
"Type": 0
}
],
"Key": "InstanceId"
},
"ObjectTypeTitle": "CVM"
}
],
"RequestId": "fd1309c8-f198-4a6e-924c-89a80a274138"
}
}
```

5. 开发者资源

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
- Tencent Cloud SDK 3.0 for Java
- Tencent Cloud SDK 3.0 for PHP
- Tencent Cloud SDK 3.0 for Go
- Tencent Cloud SDK 3.0 for Node.js
- Tencent Cloud SDK 3.0 for .NET
- Tencent Cloud SDK 3.0 for C++

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

查询任务

最近更新时间：2026-07-10 16:49:04

1. 接口描述

接口请求域名： cfg.intl.tencentcloudapi.com 。

查询任务

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeTask。
Version	是	String	公共参数，本接口取值：2021-08-20。
Region	否	String	公共参数，此参数为可选参数。
TaskId	是	Integer	任务ID

3. 输出参数

参数名称	类型	描述
Task	Task	任务信息
ReportInfo	TaskReportInfo	任务对应的演练报告信息，null表示未导出报告 注意：此字段可能返回 null，表示取不到有效值。

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

4. 示例

示例1 查询演练任务

输入示例

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

{
"TaskId": 6834
}
```

输出示例

```
{
"Response": {
"ReportInfo": null,
"RequestId": "68da5328-ff1f-4320-92c0-a67e65c8a298",
"Task": {
"AlarmPolicy": [],
"ApmServiceList": [
{
"InstanceId": "test-ins",
"RegionId": 1,
"ServiceNameList": [
"test-service"
]
}
],
"ApplicationId": "",
"ApplicationName": "",
"Tags": [],
"TaskCreateTime": "2023-10-09 10:55:18",
"TaskDescription": "游戏登录服跨AZ容灾演练",
"TaskEndTime": null,
```

```
"TaskExpect": null,
"TaskGroups": [
{
"ObjectTypeId": 1,
"TaskGroupActions": [
{
"ActionApiType": 1,
"ActionAttribute": 1,
"ActionId": 1,
"ActionRisk": "高风险",
"ActionTitle": "关机（测试）",
"ActionType": "平台",
"IsExecuteRedo": false,
"TaskGroupActionCreateTime": "2023-10-09 10:55:18",
"TaskGroupActionCustomConfiguration": "{\"force\": \"否\", \"destIp\": \"\", \"ex
cludeIp\": \"\", \"localPort\": \"\", \"netPercent\": 8, \"remotePort\": \"\",
\"excludePort\": \"\", \"netInterface\": \"eth0\"}",
"TaskGroupActionExecuteId": null,
"TaskGroupActionExecuteTime": null,
"TaskGroupActionGeneralConfiguration": "{\"AliasTitle\": \"\", \"PreTimeWait\":
0, \"ActionTimeout\": 1800, \"AfterTimeWait\": 0}",
"TaskGroupActionId": 13785,
"TaskGroupActionOrder": 1,
"TaskGroupActionRandomId": 457181,
"TaskGroupActionRecoverId": 355514,
"TaskGroupActionStatus": 2001,
"TaskGroupActionStatusType": 0,
"TaskGroupActionUpdateTime": "2023-10-09 10:55:18",
"TaskGroupInstances": []
},
{
"ActionApiType": 1,
"ActionAttribute": 2,
"ActionId": 2,
"ActionRisk": "高风险",
"ActionTitle": "开机",
"ActionType": "平台",
"IsExecuteRedo": false,
"TaskGroupActionCreateTime": "2023-10-09 10:55:18",
"TaskGroupActionCustomConfiguration": "{\"force\": \"否\", \"destIp\": \"\", \"ex
cludeIp\": \"\", \"localPort\": \"\", \"netPercent\": 8, \"remotePort\": \"\",
\"excludePort\": \"\", \"netInterface\": \"eth0\"}",
"TaskGroupActionExecuteId": 457181,
"TaskGroupActionExecuteTime": null,
"TaskGroupActionGeneralConfiguration": "{\"PreTimeWait\": 0, \"ActionTimeout\": 1
800, \"AfterTimeWait\": 0}",
"TaskGroupActionId": 13786,
```

```
"TaskGroupActionOrder": 2,
"TaskGroupActionRandomId": 355514,
"TaskGroupActionRecoverId": null,
"TaskGroupActionStatus": 2001,
"TaskGroupActionStatusType": 0,
"TaskGroupActionUpdateTime": "2023-10-09 10:55:18",
"TaskGroupInstances": []
}
],
"TaskGroupCreateTime": "2023-10-09 10:55:18",
"TaskGroupDescription": "1",
"TaskGroupDiscardInstanceList": [],
"TaskGroupId": 6454,
"TaskGroupInstanceList": [
"ins-knq6h3r8",
"ins-61eitwrk",
"ins-d2e45nba"
],
"TaskGroupInstancesExecuteRule": [
{
"TaskGroupInstancesExecuteMode": 3,
"TaskGroupInstancesExecuteNum": 2,
"TaskGroupInstancesExecutePercent": 100
}
],
"TaskGroupMode": 1,
"TaskGroupOrder": 1,
"TaskGroupSelectedInstanceList": [],
"TaskGroupTitle": "1",
"TaskGroupUpdateTime": "2023-10-09 10:55:18"
}
],
"TaskId": 5417,
"TaskMode": 1,
"TaskMonitors": [
{
"InstancesIds": [
"ins-knq6h3r8",
"ins-61eitwrk",
"ins-d2e45nba"
],
"MetricChineseName": "CPU使用率",
"MetricId": 614,
"MetricName": "CpuUsage",
"TaskMonitorId": 5850,
"TaskMonitorObjectTypeId": 1,
"Unit": "%"
```

```
}
],
"TaskOwnerUin": "700000174829",
"TaskPauseDuration": 60,
"TaskPlanId": null,
"TaskPlanTitle": null,
"TaskPolicy": null,
"TaskProtectStrategy": null,
"TaskRegionId": 1,
"TaskStartTime": null,
"TaskStatus": 1001,
"TaskStatusType": 0,
"TaskSummary": null,
"TaskTag": "",
"TaskTitle": "跨AZ容灾演练",
"TaskUpdateTime": "2023-10-09 10:55:18"
}
}
}
```


示例2 无权限的调用

输入示例

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

{
"TaskId": 6834
}
```

输出示例


```
{
"Response": {
"Error": {
"Code": "UnauthorizedOperation",
"Message": "您无权操作此功能！"
},
"RequestId": "d1fc7e73-8e6d-45fa-a333-c199914883e9"
}
}
```

5. 开发者资源

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
- Tencent Cloud SDK 3.0 for Java
- Tencent Cloud SDK 3.0 for PHP
- Tencent Cloud SDK 3.0 for Go
- Tencent Cloud SDK 3.0 for Node.js
- Tencent Cloud SDK 3.0 for .NET
- Tencent Cloud SDK 3.0 for C++

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。
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。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数错误。
OperationDenied	操作被拒绝。
RequestLimitExceeded	请求的次数超过了频率限制。

ResourceInUse	资源被占用。
ResourceInsufficient	资源不足。
ResourceNotFound	资源不存在。
ResourceUnavailable	资源不可用。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

获取演练过程日志

最近更新时间：2026-07-10 16:49:03

1. 接口描述

接口请求域名： cfg.intl.tencentcloudapi.com 。

获取演练过程中的所有日志

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：DescribeTaskExecuteLogs。
Version	是	String	公共参数，本接口取值：2021-08-20。
Region	否	String	公共参数，此参数为可选参数。
TaskId	是	Integer	任务ID
Limit	是	Integer	返回的内容行数
Offset	是	Integer	日志起始的行数。

3. 输出参数

参数名称	类型	描述
LogMessage	Array of String	日志数据

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

4. 示例

示例1 示例

输入示例

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

{
"Limit": "10",
"TaskId": "387",
"Offset": "0"
}
```

输出示例

```
{
"Response": {
"RequestId": "ef03a439-2337-441b-9d53-89b618674c98",
"LogMessage": []
}
}
```

5. 开发者资源

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
- Tencent Cloud SDK 3.0 for Java
- Tencent Cloud SDK 3.0 for PHP
- Tencent Cloud SDK 3.0 for Go

- Tencent Cloud SDK 3.0 for Node.js
- Tencent Cloud SDK 3.0 for .NET
- Tencent Cloud SDK 3.0 for C++

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
AuthFailure.UnauthorizedOperation	cam未授权操作。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
OperationDenied	操作被拒绝。

查询任务列表

最近更新时间：2026-07-10 16:49:02

1. 接口描述

接口请求域名： cfg.intl.tencentcloudapi.com 。

查询任务列表

默认接口请求频率限制：100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：DescribeTaskList。
Version	是	String	公共参数，本接口取值：2021-08-20。
Region	否	String	公共参数，此参数为可选参数。
Limit	是	Integer	分页Limit
Offset	是	Integer	分页Offset
TaskTitle	否	String	演练名称
TaskTag.N	否	Array of String	标签键
TaskStatus	否	Integer	任务状态(1001 -- 未开始 1002 -- 进行中 1003 -- 暂停中 1004 -- 任务结束)
TaskStartTime	否	String	开始时间，固定格式%Y-%m-%d %H:%M:%S
TaskEndTime	否	String	结束时间，固定格式%Y-%m-%d %H:%M:%S

TaskUpdateTime	否	String	更新时间，固定格式%Y-%m-%d %H:%M:%S
Tags.N	否	Array of TagWithDescribe	标签对
Filters.N	否	Array of ActionFilter	筛选条件
TaskId.N	否	Array of Integer	演练ID
ApplicationId.N	否	Array of String	关联应用ID筛选
ApplicationName.N	否	Array of String	关联应用筛选
TaskStatusList.N	否	Array of Integer	任务状态筛选--支持多选 任务状态(1001 -- 未开始 1002 -- 进行中 1003 -- 暂停中 1004 -- 任务结束)
ArchId	否	String	架构ID
ArchName	否	String	架构名称

3. 输出参数

参数名称	类型	描述
TaskList	Array of TaskListItem	无
Total	Integer	列表数量
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 查询演练列表

查询演练列表

输入示例

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

{
"Limit": "10",
"TaskTag": [
"飞扬"
],
"Offset": "0"
}
```

输出示例

```
{
"Response": {
"RequestId": "47e12dca-fa37-49b4-86e1-d7d3d7674640",
"TaskList": [
{
"TaskId": 2,
"TaskTitle": "广州六区-关机",
"TaskDescription": "广州六区-关机",
"TaskTag": "飞扬",
"TaskStatus": 1002,
"TaskCreateTime": "2021-08-14 00:37:34",
"TaskUpdateTime": "2021-09-18 19:18:28",
"TaskPreCheckStatus": 0,
"TaskPreCheckSuccess": false
}
],
"Total": 1
}
}
```

示例2 查询演练列表(含预检状态信息)

查询演练列表(含预检状态信息)

输入示例

```
POST / HTTP/1.1
Host: cfg.intl.tencentcloudapi.com
Content-Type: application/json
```

```
X-TC-Action: DescribeTaskList
<公共请求参数>

{
"Limit": "2",
"Offset": "0"
}
```

输出示例

```
{
"Response": {
"RequestId": "597dbef8-fcf3-46c2-9561-f87694052606",
"TaskList": [
{
"TaskId": 3077,
"TaskTitle": "演练预检测试",
"TaskDescription": "演练预检测试",
"TaskTag": "",
"TaskStatus": 1001,
"TaskCreateTime": "2022-09-19 12:57:22",
"TaskUpdateTime": "2022-09-19 12:57:22",
"TaskPreCheckStatus": 2,
"TaskPreCheckSuccess": false
},
{
"TaskId": 3076,
"TaskTitle": "【公有云】Mongo节点故障",
"TaskDescription": "【公有云】Mongo节点故障",
"TaskTag": "",
"TaskStatus": 1003,
"TaskCreateTime": "2022-09-19 11:09:50",
"TaskUpdateTime": "2022-09-19 12:17:49",
"TaskPreCheckStatus": 0,
"TaskPreCheckSuccess": false
}
],
"Total": 2450
}
}
```

5. 开发者资源

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
- Tencent Cloud SDK 3.0 for Java
- Tencent Cloud SDK 3.0 for PHP
- Tencent Cloud SDK 3.0 for Go
- Tencent Cloud SDK 3.0 for Node.js
- Tencent Cloud SDK 3.0 for .NET
- Tencent Cloud SDK 3.0 for C++

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
MissingParameter	缺少参数错误。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

查询经验库

最近更新时间：2026-07-10 16:49:00

1. 接口描述

接口请求域名： cfg.intl.tencentcloudapi.com 。

查询经验库

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：DescribeTemplate。
Version	是	String	公共参数，本接口取值：2021-08-20。
Region	否	String	公共参数，此参数为可选参数。
TemplateId	是	Integer	经验库ID

3. 输出参数

参数名称	类型	描述
Template	Template	经验库详情
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 查询经验

查询经验

输入示例

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

{
"TemplateId": 1953
}
```

输出示例


```
{
"Response": {
"RequestId": "e5cf31b7-d530-4bce-8836-4c227b463b31",
"Template": {
"AlarmPolicy": [],
"ApmServiceList": [],
"Tags": [],
"TemplateCreateTime": "2023-10-09 11:03:31",
"TemplateDescription": "cvm关机经验库，团队内部使用",
"TemplateGroups": [
{
"CreateTime": "2023-10-09 11:03:31",
"Description": "逻辑层故障场景编排",
"Mode": 1,
"ObjectTypeId": 1,
"Order": 1,
"TemplateGroupActions": [
{
"ActionApiType": 1,
"ActionAttribute": 1,
"ActionId": 1,
"ActionTitle": "关机（测试）",
"ActionType": "平台",
"CreateTime": "2023-10-09 11:03:31",
"CustomConfiguration": "{\"force\": \"否\", \"destIp\": \"\", \"excludeIp\":
\"\", \"localPort\": \"\", \"netPercent\": 8, \"remotePort\": \"\", \"excludePort
```

```
\": \"\", \"netInterface\": \"eth0\"}",
"ExecuteId": null,
"GeneralConfiguration": "{\"AliasTitle\": \"\", \"PreTimeWait\": 0, \"ActionTimeo
ut\": 1800, \"AfterTimeWait\": 0}",
"Order": 1,
"RandomId": 368448,
"RecoverId": 232618,
"TemplateGroupActionId": 3982,
"UpdateTime": "2023-10-09 11:03:31"
},
{
"ActionApiType": 1,
"ActionAttribute": 2,
"ActionId": 2,
"ActionTitle": "开机",
"ActionType": "平台",
"CreateTime": "2023-10-09 11:03:31",
"CustomConfiguration": "{\"force\": \"否\", \"destIp\": \"\", \"excludeIp\":
\"\", \"localPort\": \"\", \"netPercent\": 8, \"remotePort\": \"\", \"excludePort
\": \"\", \"netInterface\": \"eth0\"}",
"ExecuteId": 368448,
"GeneralConfiguration": "{\"PreTimeWait\": 0, \"ActionTimeout\": 1800, \"AfterTim
eWait\": 0}",
"Order": 2,
"RandomId": 232618,
"RecoverId": null,
"TemplateGroupActionId": 3983,
"UpdateTime": "2023-10-09 11:03:31"
}
],
"TemplateGroupId": 1659,
"Title": "逻辑层故障验证",
"UpdateTime": "2023-10-09 11:03:31"
}
],
"TemplateId": 1175,
"TemplateIsUsed": 1,
"TemplateMode": 1,
"TemplateMonitors": [
{
"MetricChineseName": "CPU使用率",
"MetricId": 614,
"MetricName": "CpuUsage",
"MonitorId": 2173,
"ObjectTypeId": 1
}
],
```

```
"TemplateOwnerUin": "700000174829",
"TemplatePauseDuration": 60,
"TemplatePolicy": null,
"TemplateRegionId": 1,
"TemplateSource": 0,
"TemplateTag": "",
"TemplateTitle": "CVM关机",
"TemplateUpdateTime": "2023-10-09 11:03:31"
}
}
}
```


5. 开发者资源

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
- Tencent Cloud SDK 3.0 for Java
- Tencent Cloud SDK 3.0 for PHP
- Tencent Cloud SDK 3.0 for Go
- Tencent Cloud SDK 3.0 for Node.js
- Tencent Cloud SDK 3.0 for .NET
- Tencent Cloud SDK 3.0 for C++

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
FailedOperation	操作失败。
InternalError	内部错误。

InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数错误。
OperationDenied	操作被拒绝。
ResourceInUse	资源被占用。
ResourceInsufficient	资源不足。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

执行任务

最近更新时间：2026-07-10 16:48:59

1. 接口描述

接口请求域名： cfg.intl.tencentcloudapi.com 。

执行任务

默认接口请求频率限制：10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：ExecuteTask。
Version	是	String	公共参数，本接口取值：2021-08-20。
Region	否	String	公共参数，此参数为可选参数。
TaskId	是	Integer	需要执行的任务ID

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 执行任务

输入示例

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

{
"TaskId": "222"
}
```

输出示例

```
{
"Response": {
"RequestId": "46924e75-a149-4130-aac0-853dbf0abea9"
}
}
```

5. 开发者资源

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
- Tencent Cloud SDK 3.0 for Java
- Tencent Cloud SDK 3.0 for PHP
- Tencent Cloud SDK 3.0 for Go
- Tencent Cloud SDK 3.0 for Node.js
- Tencent Cloud SDK 3.0 for .NET
- Tencent Cloud SDK 3.0 for C++

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
FailedOperation	操作失败。
InternalError	内部错误。
InvalidParameter	参数错误。
UnsupportedOperation	操作不支持。

执行任务动作实例

最近更新时间：2026-07-10 16:48:58

1. 接口描述

接口请求域名： cfg.intl.tencentcloudapi.com 。

触发混沌演练任务的动作，对于实例进行演练操作

默认接口请求频率限制：10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：ExecuteTaskInstance。
Version	是	String	公共参数，本接口取值：2021-08-20。
Region	否	String	公共参数，此参数为可选参数。
TaskId	是	Integer	任务ID
TaskActionId	是	Integer	任务动作ID
TaskInstanceIds.N	是	Array of Integer	任务动作实例ID
IsOperateAll	是	Boolean	是否操作整个任务
ActionType	是	Integer	操作类型：（1--启动 2--执行 3--跳过 5--重试）
TaskGroupId	是	Integer	动作组ID

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 示例

输入示例

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

{
"TaskId": "222",
"TaskActionId": "2430",
"TaskInstanceIds": [
4670,
4671,
4672
],
"IsOperateAll": true,
"ActionType": 1,
"TaskGroupId": 12
}
```

输出示例

```
{
"Response": {
"RequestId": "6549ed1a-911f-46dd-b6cd-2c02d5bd180f"
}
}
```

5. 开发者资源

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
- Tencent Cloud SDK 3.0 for Java
- Tencent Cloud SDK 3.0 for PHP
- Tencent Cloud SDK 3.0 for Go
- Tencent Cloud SDK 3.0 for Node.js
- Tencent Cloud SDK 3.0 for .NET
- Tencent Cloud SDK 3.0 for C++

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
AuthFailure.UnauthorizedOperation	cam未授权操作。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
MissingParameter	缺少参数错误。
OperationDenied	操作被拒绝。
ResourceInsufficient	资源不足。
ResourceNotFound	资源不存在。
ResourceUnavailable	资源不可用。
UnauthorizedOperation	未授权操作。
UnsupportedOperation	操作不支持。

修改任务运行状态

最近更新时间：2026-07-10 16:48:58

1. 接口描述

接口请求域名： cfg.intl.tencentcloudapi.com 。

修改任务运行状态

默认接口请求频率限制：10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：ModifyTaskRunStatus。
Version	是	String	公共参数，本接口取值：2021-08-20。
Region	否	String	公共参数，此参数为可选参数。
TaskId	是	Integer	任务ID
Status	是	Integer	任务状态, 1001--未开始 1002--进行中（执行）1003--进行中（暂停）1004--执行结束
IsExpect	否	Boolean	执行结果是否符合预期（当前扭转状态为执行结束时，需要必传此字段）
Summary	否	String	演习结论（当演习状态转变为执行结束时，需要填写此字段）
Issue	否	String	问题以及改进
Record	否	String	演练记录

IncludeRecordInReport	否	Integer	

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 示例1

输入示例

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

{
"Status": "1002",
"TaskId": "1698"
}
```

输出示例

```
{
"Response": {
"RequestId": "8e9a0777-ff96-4020-8aec-6802d8103689"
}
}
```

示例2 终止演练

输入示例

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

{
"Status": "1004",
"IsExpect": true,
"TaskId": "222",
"Summary": "演习结论"
}
```

输出示例

```
{
"Response": {
"RequestId": "e38eca72-e4ae-4a86-9696-7df399e672bd"
}
}
```

5. 开发者资源

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
- Tencent Cloud SDK 3.0 for Java
- Tencent Cloud SDK 3.0 for PHP
- Tencent Cloud SDK 3.0 for Go
- Tencent Cloud SDK 3.0 for Node.js
- Tencent Cloud SDK 3.0 for .NET
- Tencent Cloud SDK 3.0 for C++

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
FailedOperation	操作失败。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
MissingParameter	缺少参数错误。
OperationDenied	操作被拒绝。
ResourceInUse	资源被占用。
UnsupportedOperation	操作不支持。

触发护栏策略

最近更新时间：2026-07-10 16:48:57

1. 接口描述

接口请求域名： cfg.intl.tencentcloudapi.com 。

用于触发混沌演练护栏（类型为触发和恢复2种）

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：TriggerPolicy。
Version	是	String	公共参数，本接口取值：2021-08-20。
Region	否	String	公共参数，此参数为可选参数。
TaskId	是	Integer	混沌演练ID
Name	是	String	名称
Content	是	String	触发内容
TriggerType	是	Integer	触发类型，0--触发；1--恢复

3. 输出参数

参数名称	类型	描述

TaskId	Integer	演练ID
Success	Boolean	是否触发成功
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 1

1

输入示例

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

{
"TaskId": 5491,
"Name": "触发护栏测试",
"Content": "触发护栏测试内容",
"TriggerType": 0
}
```

输出示例

```
{
"Response": {
"RequestId": "1050e4e3-ef1a-4e2f-b100-c68dfd38fc75",
"Success": true,
"TaskId": 5491
}
}
```

5. 开发者资源

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
- Tencent Cloud SDK 3.0 for Java
- Tencent Cloud SDK 3.0 for PHP
- Tencent Cloud SDK 3.0 for Go
- Tencent Cloud SDK 3.0 for Node.js
- Tencent Cloud SDK 3.0 for .NET
- Tencent Cloud SDK 3.0 for C++

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	参数取值错误。
MissingParameter	缺少参数错误。
RequestLimitExceeded	请求的次数超过了频率限制。
UnauthorizedOperation	未授权操作。
UnsupportedOperation	操作不支持。

获取护栏触发日志

最近更新时间：2026-07-10 16:49:01

1. 接口描述

接口请求域名： cfg.intl.tencentcloudapi.com 。

获取护栏触发日志

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：DescribeTaskPolicyTriggerLog。
Version	是	String	公共参数，本接口取值：2021-08-20。
Region	否	String	公共参数，此参数为可选参数。
TaskId	是	Integer	演练ID
Page	是	Integer	页码
PageSize	是	Integer	页数量

3. 输出参数

参数名称	类型	描述
TriggerLogs	Array of PolicyTriggerLog	触发日志

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 1

1

输入示例

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

{
"TaskId": 5491,
"Page": 1,
"PageSize": 11
}
```

输出示例

```
{
"Response": {
"RequestId": "1bba6839-682a-4123-9728-ec3fc141235b",
"TriggerLogs": [
{
"Content": "触发护栏测试内容-恢复i",
"CreatTime": "2023-11-14 12:51:33",
"Name": "触发护栏测试-恢复",
"TaskId": 5491,
"TriggerType": 1
},
{
"Content": "触发护栏测试内容",
"CreatTime": "2023-11-14 12:37:20",
"Name": "触发护栏测试",
"TaskId": 5491,
"TriggerType": 0
}
```

```
]
}
}
```

5. 开发者资源

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
- Tencent Cloud SDK 3.0 for Java
- Tencent Cloud SDK 3.0 for PHP
- Tencent Cloud SDK 3.0 for Go
- Tencent Cloud SDK 3.0 for Node.js
- Tencent Cloud SDK 3.0 for .NET
- Tencent Cloud SDK 3.0 for C++

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
AuthFailure.UnauthorizedOperation	cam未授权操作。
FailedOperation	操作失败。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
MissingParameter	缺少参数错误。

RequestLimitExceeded	请求的次数超过了频率限制。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

从动作创建演练

最近更新时间：2026-07-10 16:49:09

1. 接口描述

接口请求域名： cfg.intl.tencentcloudapi.com 。

从动作创建演练

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：CreateTaskFromAction。
Version	是	String	公共参数，本接口取值：2021-08-20。
Region	否	String	公共参数，此参数为可选参数。
TaskActionId	是	Integer	动作ID，可从动作列表接口DescribeActionLibraryList获取
TaskInstances.N	是	Array of String	参与演练的实例ID
TaskTitle	否	String	演练名称，不填则默认取动作名称
TaskDescription	否	String	演练描述，不填则默认取动作描述
TaskActionGeneralConfiguration	否	String	动作通用参数，需要json序列化传入，可以从动作详情接口DescribeActionFieldConfigList获取，不填默认使用动作默认参数

TaskActionCustomConfiguration	否	String	动作自定义参数，需要json序列化传入，可以从动作详情接口DescribeActionFieldConfigList获取，不填默认使用动作默认参数，注意：必填参数，是没有默认值的，务必保证传入有效值
TaskPauseDuration	否	Integer	演练自动暂停时间，单位分钟, 不填则默认为60
TaskTags.N	否	Array of TagWithCreate	标签列表

3. 输出参数

参数名称	类型	描述
TaskId	Integer	创建成功的演练ID
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 根据动作创建演练

根据动作创建演练示例

输入示例

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

{
"TaskActionId": 12,
"TaskInstances": [
"ins-jjphrwng"
],
```

```
"TaskTitle": "CVM服务器空操作",
"TaskDescription": "创建演练测试"
}
```

输出示例

```
{
"Response": {
"RequestId": "a3875789-7d89-4bf1-8763-e9b46c6459d7",
"TaskId": 10959
}
}
```

5. 开发者资源

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
- Tencent Cloud SDK 3.0 for Java
- Tencent Cloud SDK 3.0 for PHP
- Tencent Cloud SDK 3.0 for Go
- Tencent Cloud SDK 3.0 for Node.js
- Tencent Cloud SDK 3.0 for .NET
- Tencent Cloud SDK 3.0 for C++

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

经验库相关接口
查询经验库列表

最近更新时间：2026-07-10 16:48:56

1. 接口描述

接口请求域名： cfg.intl.tencentcloudapi.com 。

查询经验库列表

默认接口请求频率限制：100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：DescribeTemplateList。
Version	是	String	公共参数，本接口取值：2021-08-20。
Region	否	String	公共参数，此参数为可选参数。
Limit	是	Integer	分页Limit, 最大值100
Offset	是	Integer	分页Offset
Title	否	String	演练名称
Tag.N	否	Array of String	标签键
IsUsed	否	Integer	状态，1---使用中， 2---停用

Tags.N	否	Array of TagWithDescribe	标签对
TemplateSource	否	Integer	经验来源 0-自建 1-专家推荐
TemplateIdList.N	否	Array of Integer	经验ID
Filters.N	否	Array of ActionFilter	过滤参数

3. 输出参数

参数名称	类型	描述
TemplateList	Array of TemplateListItem	经验库列表
Total	Integer	列表数量
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 经验库列表

查询经验库列表

输入示例

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

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

输出示例

```
{
"Response": {
"RequestId": "0c06dc9d-1e90-4062-a038-74abf2bbd43d",
"TemplateList": [
{
"TemplateId": 511,
"TemplateTitle": "跨AZ容灾",
"TemplateDescription": "跨AZ容灾场景验证",
"TemplateTag": "资源归属:研发组",
"TemplateIsUsed": 1,
"TemplateCreateTime": "2021-10-12 17:28:13",
"TemplateUpdateTime": "2021-10-12 17:48:16",
"TemplateUsedNum": 0
},
{
"TemplateId": 509,
"TemplateTitle": "CPU高负载",
"TemplateDescription": "CPU高负载故障场景验证",
"TemplateTag": "资源归属:研发组",
"TemplateIsUsed": 2,
"TemplateCreateTime": "2021-09-28 15:38:00",
"TemplateUpdateTime": "2021-10-11 16:26:45",
"TemplateUsedNum": 0
}
],
"Total": 2
}
}
```


示例2 查询经验列表

查询经验列表

输入示例

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

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

输出示例


```
{
"Response": {
"RequestId": "b96d80d2-51b4-4f0f-8501-90b5ed5f0cfc",
"TemplateList": [
{
"TemplateCreateTime": "2024-07-01 15:26:46",
"TemplateDescription": "test-descrition",
"TemplateId": 1959,
"TemplateIsUsed": 1,
"TemplateSource": 0,
"TemplateTag": "资源归属:研发组",
"TemplateTitle": "经验库编辑",
"TemplateUpdateTime": "2024-07-01 15:48:26",
"TemplateUsedNum": 0
}
],
"Total": 22
}
}
```


5. 开发者资源

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
- Tencent Cloud SDK 3.0 for Java
- Tencent Cloud SDK 3.0 for PHP
- Tencent Cloud SDK 3.0 for Go
- Tencent Cloud SDK 3.0 for Node.js
- Tencent Cloud SDK 3.0 for .NET
- Tencent Cloud SDK 3.0 for C++

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
MissingParameter	缺少参数错误。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

数据结构

最近更新时间：2026-07-10 16:49:10

ActionFieldConfigDetail

动作动态参数返回格式

被如下接口引用：DescribeActionFieldConfigList。

名称	类型	描述
Type	String	组件类型 可选项如下： input 文本框 textarea 多行文本框 number 数值输入框 select 选择器 cascader 级联选择器 radio 单选 time 时间选择
Lable	String	组件label
Field	String	组件唯一标识， 传回后端时的key
DefaultValue	String	默认值

Config	String	支持配置项如下,可根据需要选择配置项，不需要配置是设置空{}:
		
		{
		
		placeholder: string (占位符)
		
		tooltip: string (提示信息)
		
		reg: RegExp (对输入内容格式进行正则校验的规则)
		
		max: number (对于输入框，限制最大输入字符数，对于数值输入框，设置上限)
		
		min: number (对于数值输入框，设置下限)
		
		step: number (设置数值输入框的步长，默认为1)
		
		format: string (时间选择的格式，如YYYY-MM-DD表示年月日, YYYY-MM-DD HH:mm:ss 表示时分秒)
		
		separator: string[] (多行输入框的分隔符，不传或者为空时表示不分隔，直接返回用户输入的文本字符串)
		
		multiple: boolean (是否多选,对选择器和级联选择器有效)
		
		options: 选择器的选项【支持以下两种形式】
		
		直接给定选项数组 { value: string; label: string }[]
		通过调接口获取选项 { api: string(接口地址), params: string[] (接口参数,对应于参数配置的field，前端根据field对应的所有组件的输入值作为参数查询数据， 为空时在组件加载时直接请求数据) }
		}

Required	Integer	是否必填 (0 -- 否 1-- 是)
Validate	String	compute配置依赖的其他field满足的条件时通过校验（如：三个表单项中必须至少有一个填写了） [fieldName, { config: 此项保留，等待后面具体场景细化 }]
Visible	String	是否可见

ActionFieldConfigResult

动作栏位配置结果

被如下接口引用：DescribeActionFieldConfigList。

名称	类型	描述
ActionId	Integer	动作ID
ActionName	String	动作名称
ConfigDetail	Array of ActionFieldConfigDetail	动作对应的栏位配置详情

ActionFilter

动作库筛选栏位

被如下接口引用：DescribeActionLibraryList, DescribeTaskList, DescribeTemplateList。

名称	类型	必选	描述
Keyword	String	是	关键字
Values	Array of String	是	搜索内容值

ActionLibraryListResult

动作库数据列表

被如下接口引用：DescribeActionLibraryList。

名称	类型	描述
ActionName	String	动作名称
Desc	String	动作描述
ActionType	String	动作类型。范围：["平台","自定义"]
CreateTime	String	创建时间
Creator	String	创建人
UpdateTime	String	更新时间
RiskDesc	String	动作风险描述
ActionId	Integer	动作ID
AttributeId	Integer	动作属性（1：故障 2：恢复）
RelationActionId	Integer	关联的动作ID
ActionCommand	String	操作命令
ActionCommandType	Integer	动作类型（0 -- tat 1 -- 云API）
ActionContent	String	自定义动作的参数，json string
ResourceType	String	二级分类
ActionDetail	String	动作描述
IsAllowed	Boolean	是否允许当前账号使用
ActionBestCase	String	最佳实践案例的链接地址
ObjectType	String	对象类型
MetricIdList	Array of Integer	监控指标ID列表
IsNewAction	Boolean	是否是新动作
ObjectTypeId	Integer	对象类型ID

ApmServiceInfo

应用性能监控产品中应用信息

被如下接口引用：DescribeTask, DescribeTemplate。

名称	类型	必选	描述
InstanceId	String	是	业务ID
ServiceNameList	Array of String	是	应用名称
RegionId	Integer	否	地域ID

DescribePolicy

查询-保护策略

被如下接口引用：DescribeTask。

名称	类型	必选	描述
TaskPolicyIdList	Array of String	是	保护策略ID列表
TaskPolicyStatus	String	是	保护策略状态 枚举值： • 已触发： 表示已触发护栏策略 • 未触发： 表示未触发护栏策略 • 已恢复： 表示护栏策略已恢复
TaskPolicyRule	String	是	策略规则
TaskPolicyDealType	Integer	是	护栏策略生效处理策略 1:顺序执行，2:暂停

ObjectType

对象类型

被如下接口引用：DescribeObjectTypeList。

名称	类型	描述
ObjectTypeId	Integer	对象类型ID
ObjectTypeTitle	String	对象类型名称
ObjectTypeLevelOne	String	对象类型第一级
ObjectTypeParams	ObjectTypeConfig	对象类型参数
ObjectTypeJsonParse	ObjectTypeJsonParse	tke接口json解析规则，null不需要解析
ObjectHasNewAction	Boolean	是否包含新动作
ObjectPlatformName	String	对应在平台架构图中的资源类型名称
ObjectSupportType	Integer	1：平台支持的对象 2：应用支持的部分对象
ArchLayer	Integer	1.接入层 2.逻辑层 3. 数据层
IsArchSvg	Boolean	是否支持演练生图

ObjectTypeConfig

对象类型配置

被如下接口引用：DescribeObjectTypeList。

名称	类型	描述
Key	String	主键
Fields	Array of ObjectTypeConfigFields	对象类型配置字段列表

ObjectTypeConfigFields

对象类型字段类型

被如下接口引用：DescribeObjectTypeList。

名称	类型	必选	描述
Key	String	是	instanceId
Header	String	是	实例id
Transfer	String	否	字段值是否需要转译，当不需要转译时，此字段返回null
JsonParse	String	否	tke的pod字段信息解析
Type	Integer	否	字段类型 0:str 1:list

ObjectTypeJsonParse

标准pod对象类型下拉数据的解析

被如下接口引用：DescribeObjectTypeList。

名称	类型	描述
NameSpace	String	命名空间
WorkloadName	String	工作负载名称
LanIP	String	节点IP
InstanceId	String	节点ID

PolicyTriggerLog

护栏策略触发日志

被如下接口引用：DescribeTaskPolicyTriggerLog。

名称	类型	描述
TaskId	Integer	演练ID
Name	String	名称
TriggerType	Integer	类型，0--触发，1--恢复
Content	String	内容
CreatTime	String	触发时间

ResourceOffline

资源下线

被如下接口引用：DescribeActionFieldConfigList。

名称	类型	必选	描述
ResourceId	Integer	否	资源ID
ResourceDeleteTime	String	否	资源下线时间
ResourceDeleteMessage	String	否	资源下线提示

TagWithCreate

用于传入创建、编辑标签

被如下接口引用：CreateTaskFromAction, CreateTaskFromTemplate。

名称	类型	必选	描述
TagKey	String	是	标签键
TagValue	String	是	标签值

TagWithDescribe

展示标签列表

被如下接口引用：DescribeTask, DescribeTaskList, DescribeTemplate, DescribeTemplateList。

名称	类型	必选	描述
TagKey	String	是	标签键
TagValue	String	是	标签值

Task

任务

被如下接口引用：DescribeTask。

名称	类型	必选	描述
TaskId	Integer	是	任务ID
TaskTitle	String	是	任务标题
TaskDescription	String	是	任务描述
TaskTag	String	是	自定义标签
TaskStatus	Integer	是	任务状态，1001--未开始 1002--进行中（执行）1003--进行中（暂停）1004--执行结束
TaskStatusType	Integer	是	任务结束状态，表明任务以何种状态结束: 0 -- 尚未结束，1 -- 成功，2-- 失败，3--终止
TaskProtectStrategy	String	是	保护策略 注意：此字段可能返回 null，表示取不到有效值。
TaskCreateTime	String	是	任务创建时间
TaskUpdateTime	String	是	任务更新时间
TaskGroups	Array of TaskGroup	是	任务动作组
TaskStartTime	String	是	开始时间 注意：此字段可能返回 null，表示取不到有效值。
TaskEndTime	String	是	结束时间 注意：此字段可能返回 null，表示取不到有效值。
TaskExpect	Integer	是	是否符合预期。1：符合预期，2：不符合预期 注意：此字段可能返回 null，表示取不到有效值。
TaskSummary	String	是	演习记录 注意：此字段可能返回 null，表示取不到有效值。
TaskMode	Integer	是	任务模式。1:手工执行，2:自动执行
TaskPauseDuration	Integer	是	自动暂停时长。单位分钟
TaskOwnerUin	String	是	演练创建者Uin
TaskRegionId	Integer	是	地域ID
TaskMonitors	Array of TaskMonitor	是	监控指标列表
TaskPolicy	DescribePolicy	是	保护策略 注意：此字段可能返回 null，表示取不到有效值。
Tags	Array of TagWithDescribe	否	标签列表
TaskPlanId	Integer	否	关联的演练计划ID 注意：此字段可能返回 null，表示取不到有效值。
TaskPlanTitle	String	否	关联的演练计划名称 注意：此字段可能返回 null，表示取不到有效值。
ApplicationId	String	否	关联的应用ID 注意：此字段可能返回 null，表示取不到有效值。
ApplicationName	String	否	关联的应用名称
AlarmPolicy	Array of String	否	关联的告警指标
ApmServiceList	Array of ApmServiceInfo	否	关联的APM服务
VerifyId	Integer	否	关联的隐患验证项ID 注意：此字段可能返回 null，表示取不到有效值。
PolicyDealType	Integer	否	护栏处理方式，1--顺序回滚，2--演练暂停
TaskPlanStartTime	String	否	计划开始时间

TaskPlanStartTime	String	否	注意：此字段可能返回 null，表示取不到有效值。
TaskPlanEndTime	String	否	计划结束时间 注意：此字段可能返回 null，表示取不到有效值。
TaskOrg	Array of TaskOrg	否	人员组织 注意：此字段可能返回 null，表示取不到有效值。
TaskIssue	String	否	问题和改进 注意：此字段可能返回 null，表示取不到有效值。
TaskRegionName	String	否	region信息
TaskArchId	String	否	架构ID
TaskScenario	Array of TaskTarget	否	演练场景
TaskPurpose	Array of TaskTarget	否	演练目的

TaskConfig

从经验模板创建演练时需要配置的任务参数

被如下接口引用：CreateTaskFromTemplate。

名称	类型	必选	描述
TaskGroupsConfig	Array of TaskGroupConfig	是	动作组配置，需要保证配置个数和经验中的动作组个数一致
TaskTitle	String	否	更改后的演练名称，不填则默认取经验名称
TaskDescription	String	否	更改后的演练描述，不填则默认取经验描述
TaskMode	Integer	否	演练执行模式：1----手工执行/ 2 ---自动执行，不填则默认取经验执行模式
TaskPauseDuration	Integer	否	演练自动暂停时间，单位分钟，不填则默认取经验自动暂停时间
Tags	Array of TagWithCreate	否	演练标签信息，不填则默认取经验标签
PolicyDealType	Integer	否	护栏处理方式，1--顺序回滚，2--演练暂停

TaskGroup

任务分组

被如下接口引用：DescribeTask。

名称	类型	必选	描述
TaskGroupId	Integer	是	任务动作ID
TaskGroupTitle	String	是	分组标题
TaskGroupDescription	String	是	分组描述
TaskGroupOrder	Integer	是	任务分组顺序
ObjectTypeId	Integer	是	对象类型ID
TaskGroupCreateTime	String	是	任务分组创建时间
TaskGroupUpdateTime	String	是	任务分组更新时间
TaskGroupActions	Array of TaskGroupAction	是	动作分组动作列表
TaskGroupInstanceList	Array of String	是	实例列表
TaskGroupMode	Integer	是	执行模式。1 --- 顺序执行，2 --- 阶段执行
TaskGroupDiscardInstanceList	Array of String	否	不参演的实例列表
TaskGroupSelectedInstanceList	Array of String	否	参演实例列表
TaskGroupInstancesExecuteRule	Array of TaskGroupInstancesExecuteRules	否	机器选取规则

TaskGroupAction

任务分组动作

被如下接口引用：DescribeTask。

名称	类型	必选	描述
TaskGroupActionId	Integer	是	任务分组动作ID
TaskGroupInstances	Array of TaskGroupInstance	是	任务分组动作实例列表
ActionId	Integer	是	动作ID
TaskGroupActionOrder	Integer	是	分组动作顺序
TaskGroupActionGeneralConfiguration	String	是	分组动作通用配置
TaskGroupActionCustomConfiguration	String	是	分组动作自定义配置
TaskGroupActionStatus	Integer	是	分组动作状态 枚举值： • 2001： 未开始 • 2002： 待执行 • 2003： 执行中 • 2004： 执行结束
TaskGroupActionCreateTime	String	是	动作分组创建时间
TaskGroupActionUpdateTime	String	是	动作分组更新时间
ActionTitle	String	是	动作名称
TaskGroupActionStatusType	Integer	是	状态类型: 0 -- 无状态，1 -- 成功，2-- 失败，3--终止，4--跳过
TaskGroupActionRandomId	Integer	是	RandomId
TaskGroupActionRecoverId	Integer	是	RecoverId
TaskGroupActionExecuteId	Integer	是	ExecuteId
ActionApiType	Integer	否	调用api类型，0:tat, 1:云api
ActionAttribute	Integer	否	1:故障，2:恢复
ActionType	String	否	动作类型：平台、自定义
IsExecuteRedo	Boolean	否	是否可重试
ActionRisk	String	否	动作风险级别
TaskGroupActionExecuteTime	Integer	否	动作运行时间 单位：秒
TaskGroupActionStartTime	String	否	动作开始执行时间

TaskGroupActionConfig

动作组中的动作参数

被如下接口引用：CreateTaskFromTemplate。

名称	类型	必选	描述
TaskGroupActionOrder	Integer	否	该动作在动作组中的顺序，从1开始，不填或填错将匹配不到经验中要修改参数的动作
TaskGroupActionGeneralConfiguration	String	否	动作通用参数，需要json序列化传入，可以从查询经验详情接口获取，不填默认使用经验中动作参数
TaskGroupActionCustomConfiguration	String	否	动作自定义参数，需要json序列化传入，可以从查询经验详情接口获取，不填默认使用经验中动作参数

TaskGroupConfig

动作组的配置项

被如下接口引用：CreateTaskFromTemplate。

名称	类型	必选	描述
TaskGroupInstances	Array of String	是	动作组所关联的实例对象 CVM ins-xxx MySQL cdb-xxx CLB lb-xxx Redis crs-xxx NAT网关 nat-xxx 专线-独享专用通道 dcx-xxx 标准集群普通节点 {"ClusterId":"cls-xxx","InstanceId":"ins-xxx","LanIP":"1.1.1.1"} 标准集群Pod {"ClusterId":"cls-xxx","PodName":"podname","NodeName":"1.1.1.1","NameSpace":"ns","Workload":"workload"} TDSQL-MySQL(InnoDB) tdsqlshard-xxx TDSQL-C cynosdbmysql-xxx VPC子网 subnet-xxxx CKafka ckafka-xxx MariaDB tdsql-xxxx PostgreSQL postgres-xxx 云原生网关 gateway-xxx 标准集群超级节点 {"ClusterId":"cls-xxx","InstanceId":"eklet-xxx","LanIP":"1.1.1.1,"NodePoolId":"np-xxx"} Serverless集群超级节点 {"ClusterId":"cls-xxxx","InstanceId":"eklet-xxxx","LanIP":"1.1.1.1"} Elasticsearch集群 es-xxxx RabbitMQ amqp-xxxx
TaskGroupTitle	String	否	动作组标题，不填默认取经验中的动作组名称
TaskGroupDescription	String	否	动作组描述，不填默认取经验中的动作组描述
TaskGroupMode	Integer	否	动作执行模式。1 --- 顺序执行，2 --- 阶段执行，不填默认取经验中的动作组执行模式
TaskGroupActionsConfig	Array of TaskGroupActionConfig	否	动作组中的动作参数，不填默认使用经验中的动作参数，配置时可以只指定想要修改参数的动作

TaskGroupInstance

任务分组动作实例

被如下接口引用：DescribeTask。

名称	类型	必选	描述
TaskGroupInstanceId	Integer	是	实例ID
TaskGroupInstanceObjectId	String	是	实例ID
TaskGroupInstanceStatus	Integer	是	实例动作执行状态 枚举值： • 3001： 未开始 • 3002： 执行中 • 3003： 执行结束 • 3004： 准备中
TaskGroupInstanceCreateTime	String	是	实例创建时间
TaskGroupInstanceUpdateTime	String	是	实例更新时间
TaskGroupInstanceStatusType	Integer	是	状态类型: 0 -- 无状态，1 -- 成功，2-- 失败，3--终止，4--跳过
TaskGroupInstanceStartTime	String	是	执行开始时间
TaskGroupInstanceEndTime	String	是	执行结束时间
TaskGroupInstanceIsRedo	Boolean	否	实例是否可重试
TaskGroupInstanceExecuteTime	Integer	否	动作实例执行时间 单位：秒

TaskGroupInstancesExecuteRules

机器选取规则

被如下接口引用：DescribeTask。

名称	类型	必选	描述
TaskGroupInstancesExecuteMode	Integer	否	实例选取模式 枚举值： • 1： 全部注入 • 2： 随机选取指定比例注入 • 3： 随机选取指定数量注入
TaskGroupInstancesExecutePercent	Integer	否	按比例选取模式下选取比例
TaskGroupInstancesExecuteNum	Integer	否	按数量选取模式下选取数量

TaskListItem

任务列表信息

被如下接口引用：DescribeTaskList。

名称	类型	描述
TaskId	Integer	任务ID
TaskTitle	String	任务标题
TaskDescription	String	任务描述
TaskTag	String	任务标签
TaskStatus	Integer	任务状态(1001 -- 未开始 1002 -- 进行中 1003 -- 暂停中 1004 -- 任务结束)
TaskCreateTime	String	任务创建时间
TaskUpdateTime	String	任务更新时间
TaskPreCheckStatus	Integer	0--未开始，1--进行中，2--已完成
TaskPreCheckSuccess	Boolean	环境检查是否通过
TaskExpect	Integer	演练是否符合预期 1-符合预期 2-不符合预期
ApplicationId	String	关联应用ID
ApplicationName	String	关联应用名称
VerifyId	Integer	验证项ID
TaskStatusType	Integer	状态类型: 0 -- 无状态，1 -- 成功，2-- 失败，3--终止
ArchId	String	架构ID
ArchName	String	架构名称
TaskSource	Integer	来源
Tags	Array of TagWithDescribe	云资源标签列表

TaskMonitor

监控指标

被如下接口引用：DescribeTask。

名称	类型	必选	描述
TaskMonitorId	Integer	是	演练监控指标ID
MetricId	Integer	是	监控指标ID
TaskMonitorObjectTypeId	Integer	是	监控指标对象类型ID
MetricName	String	是	指标名称
InstancesIds	Array of String	是	实例ID列表
MetricChineseName	String	是	中文指标
Unit	String	是	单位

TaskOrg

演练人员组织

被如下接口引用：DescribeTask。

名称	类型	必选	描述
TaskRole	String	否	演练角色
TaskOperator	String	否	负责人

TaskReportInfo

演练报告状态信息

被如下接口引用：DescribeTask。

名称	类型	描述
Stage	Integer	0--未开始，1--正在导出，2--导出成功，3--导出失败
CreateTime	String	创建时间
ExpirationTime	String	有效期截止时间
Expired	Boolean	是否有效
CosUrl	String	演练报告cos文件地址
Log	String	演练报告导出日志 注意：此字段可能返回 null，表示取不到有效值。
ArchiveStage	Integer	0--未开始，1--正在归档，2--归档成功，3--归档失败
ArchiveTime	String	归档时间
ArchiveUuid	String	归档ID

TaskTarget

演练目标

被如下接口引用：DescribeTask, DescribeTemplate。

名称	类型	必选	描述
TargetId	Integer	否	目标标签ID
TargetDesc	String	否	目标描述
Type	Integer	否	1:演练场景 2:演练目标
Source	Integer	否	1:平台 2:用户个人
TargetStatus	Integer	否	目标标签是否已被删除 枚举值： • 0： 未删除 • 1： 已删除

Template

经验库

被如下接口引用：DescribeTemplate。

名称	类型	描述
TemplateId	Integer	经验库ID
TemplateTitle	String	经验库标题
TemplateDescription	String	经验库描述
TemplateTag	String	自定义标签
TemplateIsUsed	Integer	使用状态。1 ---- 使用中，2 --- 停用
TemplateCreateTime	String	经验库创建时间
TemplateUpdateTime	String	经验库更新时间
TemplateMode	Integer	经验库模式。1:手工执行，2:自动执行
TemplatePauseDuration	Integer	自动暂停时长。单位分钟
TemplateOwnerUin	String	演练创建者Uin
TemplateRegionId	Integer	地域ID
TemplateGroups	Array of TemplateGroup	动作组
TemplateMonitors	Array of TemplateMonitor	监控指标
TemplatePolicy	TemplatePolicy	护栏监控 注意：此字段可能返回 null，表示取不到有效值。
Tags	Array of TagWithDescribe	标签列表
TemplateSource	Integer	经验来源 0-自建 1-专家推荐
ApmServiceList	Array of ApmServiceInfo	apm应用信息
AlarmPolicy	Array of String	告警指标
PolicyDealType	Integer	护栏处理方式，1--顺序回滚，2--演练暂停
TemplateScenario	Array of TaskTarget	演练场景
TemplatePurpose	Array of TaskTarget	演练目的

TemplateGroup

任务分组

被如下接口引用：DescribeTemplate。

名称	类型	必选	描述
TemplateGroupId	Integer	是	经验库动作ID
TemplateGroupActions	Array of TemplateGroupAction	是	经验库动作分组动作列表
Title	String	是	分组标题
Description	String	是	分组描述
Order	Integer	是	分组顺序
Mode	Integer	是	执行模式。1 --- 顺序执行，2 --- 阶段执行
ObjectTypeId	Integer	是	对象类型ID
CreateTime	String	是	分组创建时间
UpdateTime	String	是	分组更新时间

TemplateGroupAction

任务分组动作

被如下接口引用：DescribeTemplate。

名称	类型	必选	描述
TemplateGroupActionId	Integer	是	经验库分组动作ID
ActionId	Integer	是	动作ID
Order	Integer	是	分组动作顺序
GeneralConfiguration	String	是	分组动作通用配置
CustomConfiguration	String	是	分组动作自定义配置
CreateTime	String	是	动作分组创建时间
UpdateTime	String	是	动作分组更新时间
ActionTitle	String	是	动作名称
RandomId	Integer	是	自身随机id
RecoverId	Integer	是	恢复动作id
ExecuteId	Integer	是	执行动作id
ActionApiType	Integer	否	调用api类型，0:tat, 1:云api
ActionAttribute	Integer	否	1:故障，2:恢复
ActionType	String	否	动作类型：平台和自定义
ActionRisk	String	否	动作风险等级，1:低风险 2:中风险 3:高风险
FailurePerformance	String	否	故障表现

TemplateListItem

经验库列表信息

被如下接口引用：DescribeTemplateList。

名称	类型	描述
TemplateId	Integer	经验库ID
TemplateTitle	String	经验库标题
TemplateDescription	String	经验库描述
TemplateTag	String	经验库标签
TemplateIsUsed	Integer	经验库状态。1 -- 使用中，2 -- 停用
TemplateCreateTime	String	经验库创建时间
TemplateUpdateTime	String	经验库更新时间
TemplateUsedNum	Integer	经验库关联的任务数量
TemplateSource	Integer	经验库来源 0-自建经验 1-专家推荐

TemplateMonitor

监控指标

被如下接口引用：DescribeTemplate。

名称	类型	描述
MonitorId	Integer	pk
MetricId	Integer	监控指标ID
ObjectTypeId	Integer	监控指标对象类型ID
MetricName	String	指标名称
MetricChineseName	String	中文指标

TemplatePolicy

保护策略

被如下接口引用：DescribeTemplate。

名称	类型	描述
TemplatePolicyIdList	Array of String	保护策略ID列表
TemplatePolicyRule	String	策略规则
TemplatePolicyDealType	Integer	护栏策略生效处理策略 1:顺序执行，2:暂停

错误码

最近更新时间：2026-07-10 16:49:10

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。
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。
OperationDenied	操作被拒绝。
ResourcesSoldOut	资源售罄。