大数据智能体工作台 DataBuddy API 文档

腾讯云

文档目录

API 文档
更新历史

最近更新时间：2026-09-11 14:18:59

第 1 次发布

发布时间：2026-09-11 14:18:49

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- CreateWorkflow
- DeleteWorkflow
- GetWorkflow
- ListWorkflows
- UpdateWorkflow

新增数据结构：

- AdvancedDependencyConfig
- AlarmBrief
- AlarmGroup
- CreateWorkflowRsp
- DeleteWorkflowRsp
- DependOnBrief
- GetWorkflowRsp
- LabelBrief
- ListWorkflowsRsp
- MonitorMetricBrief
- MonitorMetricItem
- OrderBy
- ParamInfo
- ResourceGroupInfo
- TaskRetryStrategy
- TaskRunConditionRule
- TaskType

- TaskTypeNotebookExt
- TaskTypeProperty
- UpdateWorkflowRsp
- Workflow
- WorkflowAdvanceConfig
- WorkflowBaseInfo
- WorkflowBaseInfoDetail
- WorkflowBrief
- WorkflowRunBrief
- WorkflowTask
- WorkflowTaskNodeBrief
- WorkflowTriggerAdvancedConfiguration
- WorkflowTriggerConfiguration

简介

最近更新时间：2026-09-11 14:18:57

概述

大数据智能体工作台 DataBuddy 是腾讯云推出的 Agent 原生（Agent-Native）、全托管的 Data + AI 一体化数据智能平台。DataBuddy 融合数据计算与 AI 智能体能力，通过统一元数据与语义层，为企业提供从数据接入、数据工程、数据科学、数据分析到数据治理的端到端全链路能力，让大数据平台从“人操作工具”进化为“AI 工作、人把关”。

本章节介绍的腾讯云 DataBuddy API 接口均为 API 3.0 接口。

您可以调用 API 对腾讯云 DataBuddy 进行操作，例如创建与管理工作空间、创建与变配计算资源、管理数据源连接、提交与查询 SQL 任务、创建与运行工作流、查询任务运行实例与运行状态、管理数据目录中的 Catalog/Schema/表/Volume 等资产、注册与部署模型服务、查询数据质量与治理结果等，从而将 DataBuddy 的能力集成到您自有的调度系统、运维平台或业务应用中。

术语表

腾讯云 DataBuddy API 接口的常见术语请参见下表：

术语	描述
工作空间（Workspace）	DataBuddy 的顶层资源与权限隔离容器。用于承载开发文件、任务、计算资源、成员与配置，通常按团队、项目或环境划分。调用大部分业务接口时需指定所属工作空间。
地域（Region）	物理数据中心的位置，决定数据存储与计算资源的物理位置。工作空间创建后地域不可修改，调用 API 时需通过公共参数 `Region` 指定。
数据目录（Catalog）	三级命名空间（`Catalog.Schema.Table`）中的最顶层容器。用于统一管理跨工作空间的元数据、权限与数据资产，是数据治理与访问控制的核心边界。
Schema	数据目录下的二级命名空间。用于对表、视图、函数等对象进行逻辑分组与隔离，通常对应业务域、项目或环境。
表（Table）	结构化数据的基本存储单元。包含明确的列定义与数据类型，可通过接口进行元数据查询与管理。
卷（Volume）	面向非结构化及半结构化数据（如图片、音视频、PDF、JSON、Parquet 文件等）的存储单元。位于 Schema 之下，纳入统一权限管理，供 Notebook、任务与模型训练直接访问。

术语	描述
计算资源	DataBuddy 承载计算负载的集群资源，分为作业集群、交互式集群、平台集群、实时集群四类。用于运行 SQL、Notebook、Python、Ray、数据接入等任务。
CU（Compute Unit）	计算资源的统一计量单位，1 CU ≈ 1 核 4 GB 内存。用于计算资源的规格配额设置与按量计费计量。
工作流（Workflow）	数据任务的编排与调度单元。以 DAG 形式串联 Notebook、SQL、数据接入、模型训练等节点，支持依赖管理、定时触发与告警。
任务实例	工作流中任务按调度周期或手动触发生成的一次运行记录。用于查询运行状态、日志与运行结果。
补数据（Backfill）	针对历史时间范围重新执行任务的能力。用于修复数据缺失、修正逻辑错误或初始化历史分区，支持按业务日期批量回溯。
数据源	外部系统的连接配置（数据库、数仓、消息队列、对象存储、API 等）。用于离线接入、实时接入与联邦查询，需通过连通性测试后使用。
实时接入（Realtime Ingestion）	基于 CDC（变更数据捕获）等技术的低延迟数据同步方式。将源系统的数据变更实时同步至数据湖，支撑实时数仓与实时分析场景。
离线接入（Batch Ingestion）	按周期批量将外部数据源同步到数据湖的方式。适用于对时效性要求不高的大规模数据集成。
语义模型（Semantic Model）	位于数据物理层之上的业务语义抽象层。统一定义指标、维度、实体及其关系，为 BI 分析、自然语言问数与 AI 应用提供一致的语义口径。
指标（Metric）	面向业务场景的可度量数值定义（如 GMV、留存率、活跃用户数）。包含口径、计算逻辑、维度与聚合方式，供 BI、Agent 与下游应用统一调用。
模型（Model）	机器学习模型资产，包含模型文件、依赖、签名与元数据。支持版本管理、血缘追踪与生命周期管理，可注册到模型仓库供推理或再训练使用。
模型服务（Model Service）	将注册后的模型部署为在线或批量推理服务的能力。提供 REST API、弹性扩缩容、流量路由与监控，用于将模型能力接入业务系统。
智能体（Agent）	具备规划、工具调用与记忆能力的 AI 应用单元。可对接指标、数据、模型与外部系统，完成复杂的业务任务与自动化流程。
Buddy	内嵌于平台的 AI 智能助手，覆盖数据开发、分析、治理、运维等场景。可通过自然语言协助用户写 SQL、解释代码、排查任务、生成文档等。
OBO（On Behalf Of）	以用户真实身份代理执行的权限模型。Agent 与接口调用均复用调用者在数据平台上的权限，用于防止越权访问。

术语	描述
治理标签（Governed Tags）	在数据目录中对 Catalog、Schema、表、列等对象打上的受控标签。用于数据分级分类、敏感数据识别、访问策略与合规审计。
RequestId	每次 API 调用返回的唯一请求标识。用于问题定位与工单排查，建议在业务侧日志中完整记录。

使用限制

调用腾讯云 DataBuddy API 时，请注意以下限制：

通用限制

- **鉴权与权限**：调用 API 需使用腾讯云访问密钥（SecretId/SecretKey）进行签名，并确保调用账号（主账号或子账号）已获得对应的 CAM 策略授权及 DataBuddy 工作空间内的成员角色权限。接口一律按调用者的真实身份进行权限校验，不支持越权访问其他账号或未授权工作空间下的资源。
- **密钥管理**：访问密钥仅通过环境变量或密钥管理服务读取，请勿硬编码在代码、配置文件或前端页面中；
- **地域限制**：API 通过公共参数 `Region` 指定地域，仅支持当前账号已开通 DataBuddy 服务的地域。跨地域资源不可互相操作，工作空间的地域在创建后不可修改。
- **调用频率**：接口存在调用频率限制，超出后会返回请求频率超限的错误码。建议对轮询类接口（如查询任务运行状态）采用退避重试策略，避免高频空转。
- **分页限制**：列表类接口通过 `Offset`、`Limit` 分页返回，单次返回条数存在上限。请通过循环分页获取完整数据，不要依赖单次全量返回。

API 快速入门

您可以使用 API Explorer 工具在线调用 API。

本文以**创建工作流并查看工作流信息** 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，在左侧产品列表中选择**大数据智能体工作台 DataBuddy** 。
2. 调用 ListWorkspaces，查询当前地域下的工作空间列表，获取目标工作空间的 `WorkspaceId`。后续所有工作流接口均需传入该参数。
3. 调用 CreateWorkflow，在指定工作空间下创建工作流。必填参数为 `WorkspaceId` 与 `BaseInfo`（其中 `WorkflowName` 必填，且在工作空间内唯一）；如需定时调度，可通过 `Trigger` 配置 Cron 表达式。接口返回新建工作流的 `WorkflowId`。
4. 调用 ListWorkflows，分页查询工作空间下的工作流列表，确认新建的工作流已生效。支持通过 `WorkflowNameKeyword` 按名称模糊搜索，并通过 `PageNumber`、`PageSize` 分页。
5. 调用 GetWorkflow，传入 `WorkspaceId` 与 `WorkflowId`，查看该工作流的完整定义（基本信息、任务节点、调度、告警、参数、标签等）。

6. 完成上述步骤后，您可以继续调用 RunWorkflow 运行工作流，并通过 ListWorkflowRuns、GetWorkflowRun 查询运行记录与运行状态。

请求示例

创建工作流：

```
{
"WorkspaceId": "ws-xxxxxxxx",
"BaseInfo": {
"WorkflowName": "daily_etl_pipeline",
"Description": "每日 ETL 主流程"
},
"Trigger": [
{
"Type": "CRON",
"CronExpression": "0 0 2 * * ?"
}
]
}
```

返回结果：

```
{
"Response": {
"Data": {
"WorkflowId": "wf-xxxxxxxx"
},
"RequestId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}
}
```

相关接口

工作流全生命周期涉及的接口如下：

场景	接口
创建 / 更新 / 删除工作流	CreateWorkflow、UpdateWorkflow、DeleteWorkflow
查询工作流列表 / 详情	ListWorkflows、GetWorkflow
运行 / 重跑 / 终止	RunWorkflow、RerunWorkflowRun、KillWorkflowRun
查询运行记录列表 / 详情	ListWorkflowRuns、GetWorkflowRun
查询任务运行列表 / 详情	ListWorkflowTaskRuns、GetWorkflowTaskRun

API 概览

最近更新时间：2026-09-11 14:18:59

工作流相关接口

接口名称	接口功能	频率限制（次/秒）
CreateWorkflow	创建工作流	20
ListWorkflows	查询工作流列表	20
UpdateWorkflow	更新工作流	20
DeleteWorkflow	删除工作流	20
GetWorkflow	获取工作流详细信息	20

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

调用方式
请求结构

最近更新时间：2026-09-11 14:18:57

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

接入地域	域名
亚太东北（首尔）	databuddy.ap-seoul.tencentcloudapi.com
亚太东北（东京）	databuddy.ap-tokyo.tencentcloudapi.com
美国东部（弗吉尼亚）	databuddy.na-ashburn.tencentcloudapi.com
美国西部（硅谷）	databuddy.na-siliconvalley.tencentcloudapi.com
南美地区（圣保罗）	databuddy.sa-saopaulo.tencentcloudapi.com
欧洲地区（法兰克福）	databuddy.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）	databuddy.ap-shanghai-fsi.tencentcloudapi.com
华南地区（深圳金融）	databuddy.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：2026-09-11 14:18:57

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

公共参数的具体内容会因您使用的签名方法版本不同而有所差异。

使用签名方法 v3 的公共参数

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

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

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

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

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

HTTP GET 请求结构示例：

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

Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2
```

```
018-10-09/cvm/tc3_request, SignedHeaders=content-type;host, Signature=5da7a33f69
93f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
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********************************/2
018-05-30/cvm/tc3_request, SignedHeaders=content-type;host, Signature=582c400e06
b5924a6f2b5d7d672d79c15b13162d9279b0855cfba6789a8edb4c
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********************************/2
018-05-30/cvm/tc3_request, SignedHeaders=content-type;host, Signature=582c400e06
b5924a6f2b5d7d672d79c15b13162d9279b0855cfba6789a8edb4c
Content-Type: multipart/form-data; boundary=58731222010402
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1527672334
```

```
X-TC-Region: ap-guangzhou

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

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

10
--58731222010402--
```

使用签名方法 v1 的公共参数

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

参数名称	类型	必选	描述
Action	String	是	操作的接口名称。取值参考接口文档中输入参数章节关于公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。
Region	String	-	地域参数，用来标识希望操作哪个地域的数据。接口接受的地域取值参考接口文档中输入参数公共参数 Region 的说明。**注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。**
Timestamp	Integer	是	当前 UNIX 时间戳，可记录发起 API 请求的时间。例如1529223702，如果与当前时间相差过大，会引起签名过期错误。
Nonce	Integer	是	随机正整数，与 Timestamp 联合起来，用于防止重放攻击。
SecretId	String	是	在 云API密钥 上申请的标识身份的 SecretId，一个 SecretId 对应唯一的 SecretKey ，而 SecretKey 会用来生成请求签名 Signature。
Signature	String	是	请求签名，用来验证此次请求的合法性，需要用户根据实际的输入参数计算得出。具体计算方法参见 文档。
Version	String	是	操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。

参数名称	类型	必选	描述
SignatureMethod	String	否	签名方式，目前支持 HmacSHA256 和 HmacSHA1。只有指定此参数为 HmacSHA256 时，才使用 HmacSHA256 算法验证签名，其他情况均使用 HmacSHA1 验证签名。
Token	String	否	即 安全凭证服务 所颁发的临时安全凭证中的 Token，使用时需要将 SecretId 和 SecretKey 的值替换为临时安全凭证中的 TmpSecretId 和 TmpSecretKey。使用长期密钥时不能设置此 Token 字段。
Language	String	否	指定接口返回的语言，仅部分接口支持此参数。取值：zh-CN，en-US。zh-CN 返回中文，en-US 返回英文。

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

HTTP GET 请求结构示例：

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

Host: cvm.tencentcloudapi.com
```

HTTP POST 请求结构示例：

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

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

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

地域列表

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

地域	取值
华北地区（北京）	ap-beijing
西南地区（重庆）	ap-chongqing
华南地区（广州）	ap-guangzhou
亚太东南（雅加达）	ap-jakarta
亚太东北（首尔）	ap-seoul
华东地区（上海）	ap-shanghai
亚太东南（新加坡）	ap-singapore

签名方法 v3

最近更新时间：2026-09-11 14:18:58

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

推荐使用 API Explorer　　点击调试

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

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

为什么要进行签名

签名通过以下方式帮助保护请求：

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

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

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

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

申请安全凭证

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

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

申请安全凭证的具体步骤如下：

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

签名版本 v3 签名过程

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

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

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

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

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

```
curl -X POST https://cvm.tencentcloudapi.com \
-H "Authorization: TC3-HMAC-SHA256 Credential=AKID********************************
**/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host;x-tc-action, Sign
ature=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": "instanc
e-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"" GET 请求，则为 URL 中问号（?）后面的字符串内容，例如：Limit=10&Of 注意：CanonicalQueryString 需要参考 RFC3986 进行 URLEncode 编 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 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 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:describeinsta` 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 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
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 CanonicalHe 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：

字段名称	解释
	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'` 哈希值，计算伪代码为 Lowercase(HexEncode(Hash.SHA256(RequestPayload)))，即对 H 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。对请求，RequestPayload 固定为空字符串。此示例计算结果是 `35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f13989`

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、所请求的服务串（tc3_request）。**Date 为 UTC 标准时间的日期，取值需要和公共参数 Timestamp 换算的 UTC 标准时间日期一致**；service 为产品名，必须与调域名一致。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)))。此结果是 7019a55be8395899b900fb5564e4200d984910f34794a27cb3fb7d10f

注意：

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

根据以上规则，示例中得到的待签名字符串如下：

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

3. 计算签名

1）计算派生签名密钥，伪代码如下：

```
SecretKey = "********************************"
SecretDate = HMAC_SHA256("TC3" + SecretKey, Date)
```

```
SecretService = HMAC_SHA256(SecretDate, Service)
SecretSigning = HMAC_SHA256(SecretService, "tc3_request")
```

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

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

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

2）计算签名，伪代码如下：

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

此示例计算结果是 10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f 。

4. 拼接 Authorization

按如下格式拼接 Authorization：

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

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

根据以上规则，示例中得到的值为：

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

最终完整的调用信息如下：

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

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

注意：

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

签名演示

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

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

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

- Signature Demo

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

Java

```
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_KE
Y");
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\\u
540d\"], \"Name\": \"instance-name\"}]}";
String hashedRequestPayload = sha256Hex(payload);
String canonicalRequest = httpRequestMethod + "\n" + canonicalUri + "\n" + canon
icalQueryString + "\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, strin
gToSign)).toLowerCase();
System.out.println(signature);

// ************* 步骤 4：拼接 Authorization *************
String authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + crede
ntialScope + ", "
+ "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, ac
tion.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")).hex
digest()
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.sha
256).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, payloa
d)
fmt.Println(curl)
}
```

PHP

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

// step 1: build canonical request string
$httpRequestMethod = "POST";
```

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

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

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

```
$signature = hash_hmac("SHA256", $stringToSign, $secretSigning);
echo $signature.PHP_EOL;

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

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

Ruby

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

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

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

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

puts canonical_request

# ************* 步骤 2：拼接待签名字符串 *************
```

```
credential_scope = date + '/' + service + '/' + 'tc3_request'
hashed_request_payload = Digest::SHA256.hexdigest(canonical_request)
string_to_sign = [
algorithm,
timestamp.to_s,
credential_scope,
hashed_request_payload,
].join("\n")
puts string_to_sign

# ************* 步骤 3：计算签名 *************
digest = OpenSSL::Digest.new('sha256')
secret_date = OpenSSL::HMAC.digest(digest, 'TC3' + secret_key, date)
secret_service = OpenSSL::HMAC.digest(digest, secret_date, service)
secret_signing = OpenSSL::HMAC.digest(digest, secret_service, 'tc3_request')
signature = OpenSSL::HMAC.hexdigest(digest, secret_signing, string_to_sign)
puts signature

# ************* 步骤 4：拼接 Authorization *************
authorization = "#{algorithm} Credential=#{secret_id}/#{credential_scope}, Signe
dHeaders=#{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, M
idpointRounding.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_req
uest"));
byte[] signatureBytes = HmacSHA256(secretSigning, Encoding.UTF8.GetBytes(stringT
oSign));
string signature = BitConverter.ToString(signatureBytes).Replace("-", "").ToLowe
r();
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).AddSecond
```

```
s(1551113065);
string requestPayload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u
547d\\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\\u5
40d\"], \"Name\": \"instance-name\"}]}"

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

const canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
```

```
+ signedHeaders + "\n"
+ hashedRequestPayload
console.log(canonicalRequest)

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

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

// ************* 步骤 4：拼接 Authorization *************
const authorization = algorithm + " " +
"Credential=" + SECRET_ID + "/" + credentialScope + ", " +
"SignedHeaders=" + signedHeaders + ", " +
"Signature=" + signature
console.log(authorization)

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

```
}
main()
```

C++

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

using namespace std;

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

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

```
string sha256Hex(const string &str)
{
char buf[3];
unsigned char hash[SHA256_DIGEST_LENGTH];
SHA256_CTX sha256;
SHA256_Init(&sha256);
SHA256_Update(&sha256, str.c_str(), str.size());
SHA256_Final(hash, &sha256);
std::string NewString = "";
for(int i = 0; i < SHA256_DIGEST_LENGTH; i++)
{
snprintf(buf, sizeof(buf), "%02x", hash[i]);
NewString = NewString + buf;
}
return NewString;
}

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

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

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

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

```
HMAC_CTX_free(h);
#endif

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

return (ss.str());
}

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

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

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

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

// ************* 步骤 1：拼接规范请求串 *************
string httpRequestMethod = "POST";
string canonicalUri = "/";
string canonicalQueryString = "";
string lower = action;
std::transform(action.begin(), action.end(), lower.begin(), ::tolower);
string canonicalHeaders = string("content-type:application/json; charset=utf-8
\n")
+ "host:" + host + "\n"
+ "x-tc-action:" + lower + "\n";
string signedHeaders = "content-type;host;x-tc-action";
string payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u
540d\"], \"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" + credentialSco
pe + "\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 + "/" + crede
ntialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
cout << authorization << endl;

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

C

```
#include <ctype.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdint.h>
```

```
#include <openssl/sha.h>
#include <openssl/hmac.h>

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

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

void hmac_sha256(const char* key, int key_len,
const char* input, int input_len,
unsigned char* output, unsigned int* output_len)
{
HMAC_CTX *h;
#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX hmac;
HMAC_CTX_init(&hmac);
h = &hmac;
#else
h = HMAC_CTX_new();
#endif
```

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


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

}

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

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

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

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

// ************* 步骤 1：拼接规范请求串 *************
const char* http_request_method = "POST";
const char* canonical_uri = "/";
const char* canonical_query_string = "";
char canonical_headers[100] = {"content-type:application/json; charset=utf-8\nho
st:"};
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\\u54
7d\\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_signi
ng, &output_len);
unsigned char k_hmac_sha_sign[64] = {0};
hmac_sha256(k_signing, output_len, string_to_sign, strlen(string_to_sign), k_hma
c_sha_sign, &output_len);

char signature[128] = {0};
hex_encode(k_hmac_sha_sign, output_len, signature);
```

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

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

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

其他语言

- Lua: GitHub
- Swift: GitHub
- Dart: GitHub
- Shell(Bash): GitHub

签名失败

存在以下签名失败的错误码，请根据实际情况处理。

错误码	错误描述
AuthFailure.SignatureExpire	签名过期。Timestamp 与服务器接收到请求的时间相差不得超过五分钟。

错误码	错误描述
AuthFailure.SecretIdNotFound	密钥不存在。请到控制台查看密钥是否被禁用，是否少复制了字符或者多了字符。
AuthFailure.SignatureFailure	签名错误。可能是签名计算错误，或者签名与实际发送的内容不相符合，也有可能是密钥 SecretKey 错误导致的。
AuthFailure.TokenFailure	临时证书 Token 错误。
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）。

签名方法

最近更新时间：2026-09-11 14:18:58

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

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

推荐使用 API Explorer　　点击调试

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

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

1. 申请安全凭证

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

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

申请安全凭证的具体步骤如下：

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

注意：每个账号最多可以拥有两对 SecretId/SecretKey。

2. 生成签名串

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

假设用户的 SecretId 和 SecretKey 分别是：

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

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

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

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

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

2.1. 对参数排序

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

```
{
'Action' : 'DescribeInstances',
'InstanceIds.0' : 'ins-09dx96dg',
'Limit' : 20,
'Nonce' : 11886,
```

```
'Offset' : 0,
'Region' : 'ap-guangzhou',
'SecretId' : 'AKID*******************************',
'Timestamp' : 1465185768,
'Version': '2017-03-12',
}
```

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

2.2. 拼接请求字符串

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

然后将格式化后的各个参数用"&"拼接在一起，最终生成的请求字符串为:

```
Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=
0&Region=ap-guangzhou&SecretId=AKID*******************************&Timestamp=14
65185768&Version=2017-03-12
```

2.3. 拼接签名原文字符串

此步骤生成签名原文字符串。
签名原文字符串由以下几个参数构成:

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

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

示例的拼接结果为:

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

```
**************&Timestamp=1465185768&Version=2017-03-12
```

2.4. 生成签名串

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

具体代码如下，以 PHP 语言为例：

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

最终得到的签名串为：

```
7RAM2xfNMO9EiVTNmPg06MRnCvQ=
```

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

3. 签名串编码

生成的签名串并不能直接作为请求参数，需要对其进行 URL 编码。

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

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

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

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

4. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理。

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

5. 签名演示

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

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

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

- Signature Demo

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

最终输出的 url 可能为：`https://cvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKID**********&Signature=7RAM2xfNMO9EiVTNmPg06MRnCvQ%3D&Timestamp=1465185768&Version=2017-03-12` 。

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

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

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

Java

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

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

public static String sign(String s, String key, String method) throws Exception
{
Mac mac = Mac.getInstance(method);
SecretKeySpec secretKeySpec = new SecretKeySpec(key.getBytes(CHARSET), mac.getAl
gorithm());
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 UnsupportedEn
codingException {
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(), CHA
RSET)).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.lan
g.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.currentTimeMi
llis() / 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("TENCENTCLOU
D_SECRET_KEY"), "HmacSHA1")); // 公共参数
System.out.println(getUrl(params));
```

```
}
}
```

Python

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

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

import requests

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

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

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

if __name__ == '__main__':
endpoint = "cvm.tencentcloudapi.com"
data = {
'Action' : 'DescribeInstances',
'InstanceIds.0' : 'ins-09dx96dg',
'Limit' : 20,
```

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

Golang

```
package main

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

func main() {
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID**********************
******
secretId := os.Getenv("TENCENTCLOUD_SECRET_ID")
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 **********************
***
secretKey := os.Getenv("TENCENTCLOUD_SECRET_KEY")
params := map[string]string{
```

```
"Nonce": "11886",
"Timestamp": strconv.Itoa(1465185768),
"Region": "ap-guangzhou",
"SecretId": secretId,
"Version": "2017-03-12",
"Action": "DescribeInstances",
"InstanceIds.0": "ins-09dx96dg",
"Limit": strconv.Itoa(20),
"Offset": strconv.Itoa(0),
}

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

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

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

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

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

PHP

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

ksort($param);

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

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

```
// curl_setopt($ch, CURLOPT_URL, $url);
// $output = curl_exec($ch);
// curl_close($ch);
// echo json_decode($output);
```

Ruby

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

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

method = 'GET'
endpoint = 'cvm.tencentcloudapi.com'
data = {
'Action' => 'DescribeInstances',
'InstanceIds.0' => 'ins-09dx96dg',
'Limit' => 20,
'Nonce' => 11886,
'Offset' => 0,
'Region' => 'ap-guangzhou',
'SecretId' => secret_id,
'Timestamp' => 1465185768, # Time.now.to_i
'Version' => '2017-03-12',
}
sign = method + endpoint + '/?'
params = []
data.sort.each do |item|
params << "#{item[0]}=#{item[1]}"
```

```
end
sign += params.join('&')
digest = OpenSSL::Digest.new('sha1')
data['Signature'] = Base64.encode64(OpenSSL::HMAC.digest(digest, secret_key, 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> requestP
arams, 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>
(param, StringComparer.Ordinal);
string sigInParam = MakeSignPlainText(headers, "GET", endpoint, "/");
string sigOutParam = Sign(SECRET_KEY, sigInParam);
Console.WriteLine(sigOutParam);
}
}
```

NodeJS

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

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

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

function sort_params(params){
let strParam = "";
```

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

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

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

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

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

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

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

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

返回结果

最近更新时间：2026-09-11 14:18:58

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

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

正确返回结果

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

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

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

错误返回结果

若调用失败，其返回值示例如下为：

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

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

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

公共错误码

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

参数类型

最近更新时间：2026-09-11 14:18:58

目前腾讯云 API 3.0 输入参数和输出参数支持如下几种数据格式：

- String: 字符串。
- Integer：整型，上限为无符号64位整数。SDK 3.0 不同编程语言支持的类型有所差异，建议以所使用编程语言的最大整型定义，例如 `Golang` 的 `uint64`。
- Boolean：布尔型。
- Float：浮点型。
- Double：双精度浮点型。
- Date：字符串，日期格式。例如：2022-01-01。
- Timestamp：字符串，时间格式。例如：2022-01-01 00:00:00。
- Timestamp ISO8601：ISO 8601 是由国际标准化组织（International Organization for Standardization，ISO）发布的关于日期和时间格式的国际标准，对应国标《GB/T 7408-2005数据元和交换格式信息交换日期和时间表示法》。建议以所使用编程语言的标准库进行格式解析。例如：2022-01-01T00:00:00+08:00。
- Binary：二进制内容，需要以特定协议请求和解析。

工作流相关接口
创建工作流

最近更新时间：2026-09-11 14:18:56

1. 接口描述

接口请求域名： databuddy.tencentcloudapi.com 。

创建工作流

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：CreateWorkflow。
Version	是	String	公共参数，本接口取值：2026-07-15。
Region	是	String	公共参数，详见产品支持的 地域列表。
WorkspaceId	是	String	工作空间ID，可通过 ListWorkspaces 获取。必填 示例值：17697410068842890
BaseInfo	是	WorkflowBaseInfo	工作流基本信息。必填，其中 WorkflowName 必填且工作空间内唯一

参数名称	必选	类型	描述
Trigger.N	否	Array of WorkflowTriggerConfiguration	工作流调度配置
ParamList.N	否	Array of ParamInfo	工作流参数列表
LabelList.N	否	Array of LabelBrief	标签列表
Alarm	否	AlarmBrief	工作流告警配置
MonitorMetric	否	MonitorMetricBrief	监控指标配置。若告警条件中选择了监控告警，则本字段必填
AdvanceConfig	否	WorkflowAdvanceConfig	工作流高级设置
TaskList.N	否	Array of WorkflowTask	工作流任务列表
BundleId	否	String	BundleId，可通过 Bundle 相关接口获取 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76
BundleInfo	否	String	Bundle信息 示例值：demo
GitConfigId	否	String	Git配置ID，可通过 Git 配置相关接口获取 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76
GitBranch	否	String	Git分支信息 示例值：main

3. 输出参数

参数名称	类型	描述
Data	CreateWorkflowRsp	创建工作流响应内容 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 创建携带python任务的工作流

创建携带python任务的工作流

输入示例

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

{
"WorkspaceId": "17697410068842890",
"BaseInfo": {
"WorkflowName": "*n**g*****n*pr**"
},
"AdvanceConfig": {
"QueuingMode": "OFF",
"MaxConcurrentNum": 1
},
"TaskList": [
{
"ParamList": [],
"DependOnList": [],
"TaskName": "py_0807",
"TaskType": {
```

```
"TaskTypeName": "PYTHON",
"TaskTypePropertyList": [
{
"PropertyKey": "Source",
"PropertyValue": "5"
}
],
"RuntimePropertyList": [
{
"PropertyKey": "ResourceMode",
"PropertyValue": "1"
}
]
},
"ResourceGroupId": "res-ea***8f8",
"TaskRetryStrategy": {},
"DependOnRunCondition": "ALL_SUCCESS",
"LeftCoordinate": 50,
"TopCoordinate": 50
}
]
}
```


输出示例


```
{
"Response": {
"Data": {
"WorkflowId": "9a1670f3-14df-4869-9f38-ec31a4c4aac9"
},
"RequestId": "81772cad-5e71-4a7c-b063-a7b5db0d929a"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation	操作失败。
FailedOperation.CallThirdPartApiError	调用 <serviceName> 服务的接口 <apiName> 失败：<message>。
FailedOperation.CreateWorkflowFailed	FailedOperation.CreateWorkflowFailed
FailedOperation.LabelCountLimit	标签数量已达到最大允许限制
FailedOperation.WorkflowCountLimit	工作流数量超过10000上限
FailedOperation.WorkflowCreateLockAcquireFailed	获取工作流名称分布式锁失败
FailedOperation.WorkflowNoPermission	无操作权限
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.DuplicateTaskNameError	同一工作流内存在重名任务
InvalidParameterValue.ParamIllegalError	参数 <parameter> 不符合要求：<message>
InvalidParameterValue.TaskHookValidationFailed	引用对象不存在或已被删除
InvalidParameterValue.TaskNameContainsIllegalCharactersError	任务名包含非法字符
InvalidParameterValue.TaskNameExceedsLimitError	任务名超过128字符限制
InvalidParameterValue.WorkflowNameExists	InvalidParameterValue.WorkflowNameExists
InvalidParameterValue.WorkflowNameInvalid	InvalidParameterValue.WorkflowNameInvalid
MissingParameter	缺少参数错误。
ResourceNotFound.WorkflowNotFound	ResourceNotFound.WorkflowNotFound

查询工作流列表

最近更新时间：2026-09-11 14:18:55

1. 接口描述

接口请求域名： databuddy.tencentcloudapi.com 。

查询工作流列表

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：ListWorkflows。
Version	是	String	公共参数，本接口取值：2026-07-15。
Region	是	String	公共参数，详见产品支持的 地域列表。
WorkspaceId	是	String	工作空间ID，可通过 ListWorkspaces 获取。必填 示例值：17697410068842890
PageNumber	否	Integer	分页页码，从 1 开始。非必填，默认 1 示例值：1
PageSize	否	Integer	每页大小。非必填，默认 10，取值范围 [10, 200] 示例值：10

参数名称	必选	类型	描述
WorkflowNameKeyword	否	String	工作流名称关键字，对 WorkflowName 做模糊匹配。非必填，单值 示例值：demo
WorkflowNames.N	否	Array of String	工作流名称，精确匹配。非必填，多选（多个值之间为 OR 关系） 示例值：["demo"]
WorkflowIds.N	否	Array of String	工作流ID，精确匹配。非必填，多选（多个值之间为 OR 关系） 示例值：["f4ec3e79-7368-44c9-8b04-031fad7f1a76"]
RunUserUins.N	否	Array of String	工作流运行人UIN，精确匹配。非必填，多选（多个值之间为 OR 关系） 示例值：["100044134096"]
LabelKeyIds.N	否	Array of String	标签名称ID，精确匹配，可通过标签相关接口获取。非必填，多选（多个值之间为 OR 关系） 示例值：["f4ec3e79-7368-44c9-8b04-031fad7f1a76"]
LabelValueIds.N	否	Array of String	标签值ID，精确匹配，可通过标签相关接口获取。非必填，多选（多个值之间为 OR 关系） 示例值：["f4ec3e79-7368-44c9-8b04-031fad7f1a76"]
QuickSelectionType	否	String	快速筛选类型。非必填，单值 对齐老云 API（wedata/2025-10-10）文档示例值： • MY_FAVORITE：我收藏的 • MY_OWNER：我负责的 • MY_AUTHORITY：我有权限 • WorkflowId：支持多个工作流ID筛选

参数名称	必选	类型	描述
			后端实现现状：当前仅 MY_FAVORITE 生效（设置 favoriteUserUin 过滤当前用户收藏），MY_OWNER / MY_AUTHORITY 暂未在 Service 层实现，传入会被忽略（按全量返回）。 示例值：MY_FAVORITE
OrderBys.N	否	Array of OrderBy	排序条件，多个之间按数组顺序表示优先级。非必填。 可排序字段白名单：CreateTime

3. 输出参数

参数名称	类型	描述
Data	ListWorkflowsRsp	查询工作流列表响应内容 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 查询工作流列表

查询工作流列表

输入示例

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

{
```

```
"WorkspaceId": "17697410068842890"
}
```


输出示例


```
{
"Response": {
"Data": {
"Items": [
{
"BundleId": "",
"BundleInfo": "",
"CreateTime": "1785835396477",
"CreateUserUin": "*******64618",
"Description": "",
"GitConfigId": "",
"LabelList": [],
"OwnerDisplayName": "we*a***************t****",
"OwnerUserName": "w***t*****e***encent.com",
"OwnerUserUin": "700002164618",
"Permission": "MANAGE",
"ResourceGroupInfoList": [],
"RunUserName": "w***ta30-dev**********m",
"RunUserUin": "700002164618",
"TaskList": [],
"Trigger": [],
"UpdateTime": "1785835396477",
"WorkflowId": "221dc674-b463-4ea7-a1ed-d76bccf4b5e6",
"WorkflowName": "new_workflow_20260804_172315",
"WorkflowRunList": []
}
],
"PageNumber": 1,
"PageSize": 10,
"TotalCount": 1176,
"TotalPageNumber": 118
},
```

```
"RequestId": "b0ba39a2-8f9d-439e-8fa7-29a40192d50d"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
FailedOperation	操作失败。
FailedOperation.WorkflowNoPermission	无操作权限
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.ListWorkflowFilterParamError	InvalidParameterValue.ListWorkflowFilterParamError
InvalidParameterValue.ParamBlankError	参数 <parameter> 不能为空。
InvalidParameterValue.ParamIllegalError	参数 <parameter> 不符合要求：<message>
MissingParameter	缺少参数错误。

更新工作流

最近更新时间：2026-09-11 14:18:54

1. 接口描述

接口请求域名： databuddy.tencentcloudapi.com 。

更新工作流

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：UpdateWorkflow。
Version	是	String	公共参数，本接口取值：2026-07-15。
Region	是	String	公共参数，详见产品支持的 地域列表。
WorkspaceId	是	String	工作空间ID，可通过 ListWorkspaces 获取。必填 示例值：17697410068842890
WorkflowId	是	String	待更新的工作流ID，可通过 ListWorkflows 获取。必填 示例值：8e0e3485-2736-4530-92ed-932e019898f1
FieldToRemoveList.N	否	Array of String	需要清空的字段名列表，用于将指定字段重置为空

参数名称	必选	类型	描述
			示例值：["demo"]
NewSetting	否	Workflow	更新后的工作流配置，仅传入需要变更的部分即可

3. 输出参数

参数名称	类型	描述
Data	UpdateWorkflowRsp	更新工作流响应内容 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 更新工作流

输入示例

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

{
"WorkspaceId": "17697410068842890",
"WorkflowId": "8e0e3485-2736-4530-92ed-932e019898f1",
"NewSetting": {
"TaskList": [
{
"ParamList": [],
"DependOnList": [],
```

```
"TaskName": "py_0807",
"TaskType": {
"TaskTypeName": "PYTHON",
"TaskTypePropertyList": [
{
"PropertyKey": "Source",
"PropertyValue": "5"
}
],
"RuntimePropertyList": [
{
"PropertyKey": "ResourceMode",
"PropertyValue": "1"
}
]
},
"ResourceGroupId": "res******8f8",
"TaskRetryStrategy": {},
"DependOnRunCondition": "ALL_SUCCESS",
"LeftCoordinate": 50,
"TopCoordinate": 50
}
]
}
}
```


输出示例


```
{
"Response": {
"Data": {
"Status": true
},
"RequestId": "516da81c-51c7-4f90-9cad-8545fb200113"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation	操作失败。
FailedOperation.CallThirdPartApiError	调用 <serviceName> 服务的接口 <apiName> 失败：<message>。
FailedOperation.UpdateWorkflowFailed	FailedOperation.UpdateWorkflowFailed
FailedOperation.WorkflowBundleNoPermission	FailedOperation.WorkflowBundleNoPermission
FailedOperation.WorkflowCreateLockAcquireFailed	获取工作流名称分布式锁失败
FailedOperation.WorkflowNoPermission	无操作权限
InvalidParameterValue	参数取值错误。
InvalidParameterValue.DuplicateTaskNameError	同一工作流内存在重名任务
InvalidParameterValue.LoopDataArrayElementCountLimit	InvalidParameterValue.LoopDataArrayElementCountLimit
InvalidParameterValue.LoopDataArrayJsonInvalid	InvalidParameterValue.LoopDataArrayJsonInvalid
InvalidParameterValue.LoopDataArrayNotJsonArray	InvalidParameterValue.LoopDataArrayNotJsonArray
InvalidParameterValue.LoopDataArrayNotJsonArrayLiteral	InvalidParameterValue.LoopDataArrayNotJsonArrayLiteral
InvalidParameterValue.LoopDataArrayValueBlank	InvalidParameterValue.LoopDataArrayValueBlank
InvalidParameterValue.LoopDataArrayValueLengthLimit	InvalidParameterValue.LoopDataArrayValueLengthLimit
InvalidParameterValue.LoopDataArrayVariableExpressionInvalid	InvalidParameterValue.LoopDataArrayVariableExpressionInvalid
InvalidParameterValue.ParamIllegalError	参数 <parameter> 不符合要求：<message>
InvalidParameterValue.TaskHookValidationFailed	引用对象不存在或已被删除
InvalidParameterValue.TaskNameContainsIllegalCharactersError	任务名包含非法字符
InvalidParameterValue.TaskNameExceedsLimitError	任务名超过128字符限制
InvalidParameterValue.WorkflowNameExists	InvalidParameterValue.WorkflowNameExists
InvalidParameterValue.WorkflowNameInvalid	InvalidParameterValue.WorkflowNameInvalid
InvalidParameterValue.WorkflowStartTimeAfterEndTimeError	工作流调度开始时间不能晚于结束时间
MissingParameter	缺少参数错误。
ResourceNotFound	资源不存在。
ResourceNotFound.WorkflowNotExist	ResourceNotFound.WorkflowNotExist
ResourceNotFound.WorkflowNotFound	ResourceNotFound.WorkflowNotFound
ResourceNotFound.WorkflowTriggerNotFound	ResourceNotFound.WorkflowTriggerNotFound

删除工作流

最近更新时间：2026-09-11 14:18:56

1. 接口描述

接口请求域名： databuddy.tencentcloudapi.com 。

删除工作流

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：DeleteWorkflow。
Version	是	String	公共参数，本接口取值：2026-07-15。
Region	是	String	公共参数，详见产品支持的 地域列表。
WorkspaceId	是	String	工作空间ID，可通过 ListWorkspaces 获取。必填 示例值：17697410068842890
WorkflowId	是	String	待删除的工作流ID，可通过 ListWorkflows 获取。必填 示例值：8b5b785c-2f9c-4adb-b36c-db264542c200

3. 输出参数

参数名称	类型	描述
Data	DeleteWorkflowRsp	删除工作流响应内容 注意：此字段可能返回 null，表示取不到有效值。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 删除工作流

输入示例

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

{
"WorkspaceId": "17697410068842890",
"WorkflowId": "8b5b785c-2f9c-4adb-b36c-db264542c200"
}
```

输出示例

```
{
"Response": {
"Data": {
"Status": true
},
"RequestId": "de368bff-f906-4d08-9255-4f313fd95006"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation	操作失败。
FailedOperation.CallThirdPartApiError	调用 <serviceName> 服务的接口 <apiName> 失败：<message>。
FailedOperation.WorkflowNoPermission	无操作权限
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。

错误码	描述
InvalidParameterValue.ParamBlankError	参数 <parameter> 不能为空。
MissingParameter	缺少参数错误。
ResourceNotFound.OneFlowResourceNoExistError	资源不存在或已被删除

获取工作流详细信息

最近更新时间：2026-09-11 14:18:55

1. 接口描述

接口请求域名： databuddy.tencentcloudapi.com 。

获取工作流详细信息

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：GetWorkflow。
Version	是	String	公共参数，本接口取值：2026-07-15。
Region	是	String	公共参数，详见产品支持的 地域列表。
WorkspaceId	是	String	工作空间ID，可通过 ListWorkspaces 获取。必填 示例值：17697410068842890
WorkflowId	是	String	工作流ID，可通过 ListWorkflows 获取。必填 示例值：8e0e3485-2736-4530-92ed-932e019898f1

3. 输出参数

参数名称	类型	描述
Data	GetWorkflowRsp	获取工作流详细信息响应内容 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 获取工作流详细信息

输入示例

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

{
"WorkspaceId": "17697410068842890",
"WorkflowId": "8e0e3485-2736-4530-92ed-932e019898f1"
}
```

输出示例

```
{
"Response": {
"Data": {
"AdvanceConfig": {
"MaxConcurrentNum": 1,
"QueuingMode": "OFF"
},
"BaseInfo": {
"CreateTime": "1786100692847",
```

```
"CreateUserUin": "700002164618",
"OwnerDisplayName": "we***a**-d**@t*n********",
"OwnerUserName": "wed**a*0*d**************",
"OwnerUserUin": "700002164618",
"RunUserUin": "700002164618",
"UpdateTime": "1786115139666",
"WorkflowId": "8e0e3485-2736-4530-92ed-932e019898f1",
"WorkflowName": "new_workflow_20260807_190451"
},
"BundleId": "",
"BundleInfo": "",
"GitConfigId": "",
"LabelList": [],
"ParamList": [
{
"ParamId": "d667305e-a68c-4661-93ea-cad9c9f1ce9c",
"ParamKey": "pppp1",
"ParamValue": "vvvv1"
}
],
"TaskList": [
{
"CreateTime": "1786100692847",
"CreateUserUin": "700002164618",
"DependOnList": [],
"DependOnRunCondition": "ALL_SUCCESS",
"LeftCoordinate": 50,
"ParamList": [],
"ResourceGroupId": "res-ea1a38f8",
"TaskId": "40d34e5c-12d9-4ac3-88ed-2f879713d4e4",
"TaskName": "py_08072",
"TaskRetryStrategy": {
"MaxRetryTimes": 3,
"RetryBetweenWaitTime": 5,
"RetryBetweenWaitTimeUnit": "SECOND",
"TaskRunFailureRetrySwitch": true,
"TaskRunTimeoutRetrySwitch": false
},
```

```
"TaskType": {
"RuntimePropertyList": [
{
"PropertyKey": "ConfigType",
"PropertyValue": "DEFAULT"
}
],
"TaskTypeName": "PYTHON",
"TaskTypePropertyList": [
{
"PropertyKey": "Source",
"PropertyValue": "5"
}
]
},
"TopCoordinate": 50,
"UpdateTime": "1786115139765"
}
],
"Trigger": [],
"WorkspaceId": "17697410068842890"
},
"RequestId": "a225ce2c-c8dd-47bc-83c1-df352a949443"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation	操作失败。
FailedOperation.WorkflowNoPermission	无操作权限
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.ParamBlankError	参数 <parameter> 不能为空。
MissingParameter	缺少参数错误。
ResourceNotFound.WorkflowNotFound	ResourceNotFound.WorkflowNotFound

数据结构

最近更新时间：2026-09-11 14:18:57

AdvancedDependencyConfig

高级依赖配置

被如下接口引用：CreateWorkflow, GetWorkflow, ListWorkflows, UpdateWorkflow。

名称	类型	必选	描述
Operator	String	否	逻辑运算符号OR / AND 注意：此字段可能返回 null，表示取不到有效值。 示例值：AND
Conditions	Array of TaskRunConditionRule	否	任务运行条件规则列表 注意：此字段可能返回 null，表示取不到有效值。

AlarmBrief

告警配置

被如下接口引用：CreateWorkflow, GetWorkflow, UpdateWorkflow。

名称	类型	必选	描述
AlarmId	String	否	告警 ID，创建时无需传入，由服务端生成 注意：此字段可能返回 null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76
AlarmMonitorType	String	否	告警的监控对象类型，如工作流、任务等，当前支持 1. WORKFLOW 2. TASK 注意：此字段可能返回 null，表示取不到有效值。 示例值：WORKFLOW

名称	类型	必选	描述
AlarmGroups	Array of AlarmGroup	否	告警组，最多 50 个 注意：此字段可能返回 null，表示取不到有效值。
DoNotDisturbWhenSkipped	Boolean	否	被跳过时免打扰，默认值 false 注意：此字段可能返回 null，表示取不到有效值。 示例值：false
DoNotDisturbWhenManuallyTerminated	Boolean	否	被手动终止时免打扰，默认值 false 注意：此字段可能返回 null，表示取不到有效值。 示例值：false
DoNotDisturbUntilTheLastRetry	Boolean	否	最后一次重试前免打扰，默认值 false 注意：此字段可能返回 null，表示取不到有效值。 示例值：false

AlarmGroup

告警组

被如下接口引用：CreateWorkflow, GetWorkflow, UpdateWorkflow。

名称	类型	必选	描述
ChannelId	String	否	通知渠道ID，可通过基础平台通知渠道相关接口获取 注意：此字段可能返回 null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76
ChannelName	String	否	通知渠道名称，可以是用户组名称或邮箱地址 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo
IsEmailChannel	Boolean	否	是否启用邮件渠道，默认值：false 注意：此字段可能返回 null，表示取不到有效值。 示例值：false

名称	类型	必选	描述
AlarmConditions	Array of String	否	一组告警条件，有 启动，成功，失败和任务超时告警 注意：此字段可能返回 null，表示取不到有效值。 示例值：["demo"]
ChannelType	Integer	否	通知渠道类型。取值：0 未指定，1 Email，2 Webhook，3 Teams，4 Slack 注意：此字段可能返回 null，表示取不到有效值。 示例值：1

CreateWorkflowRsp

CreateWorkflowRsp

被如下接口引用：CreateWorkflow。

名称	类型	必选	描述
WorkflowId	String	否	工作流ID 注意：此字段可能返回 null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76

DeleteWorkflowRsp

DeleteWorkflowRsp

被如下接口引用：DeleteWorkflow。

名称	类型	必选	描述
Status	Boolean	否	删除状态，true 表示成功 注意：此字段可能返回 null，表示取不到有效值。 示例值：false

DependOnBrief

任务依赖简要信息

被如下接口引用：CreateWorkflow, GetWorkflow, ListWorkflows, UpdateWorkflow。

名称	类型	必选	描述
TaskId	String	否	任务ID，可通过 ListWorkflowTasks 获取 注意：此字段可能返回 null，表示取不到有效值。 示例值：a4dfb8d2-cc55-4a24-8573-0091b791a927
TaskName	String	否	任务名称 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo

GetWorkflowRsp

GetWorkflowRsp

被如下接口引用：GetWorkflow。

名称	类型	必选	描述
WorkspaceId	String	否	工作空间ID 注意：此字段可能返回 null，表示取不到有效值。 示例值：17622177773248536
BaseInfo	WorkflowBaseInfoDetail	否	工作流基本信息 注意：此字段可能返回 null，表示取不到有效值。
Trigger	Array of WorkflowTriggerConfiguration	否	工作流调度配置 注意：此字段可能返回 null，表示取不到有效值。
ParamList	Array of ParamInfo	否	工作流参数列表 注意：此字段可能返回 null，表示取不到有效值。
LabelList	Array of LabelBrief	否	标签列表 注意：此字段可能返回 null，表示取不到有效值。

名称	类型	必选	描述
Alarm	AlarmBrief	否	工作流告警配置 注意：此字段可能返回 null，表示取不到有效值。
MonitorMetric	MonitorMetricBrief	否	监控指标配置 注意：此字段可能返回 null，表示取不到有效值。
AdvanceConfig	WorkflowAdvanceConfig	否	工作流高级设置 注意：此字段可能返回 null，表示取不到有效值。
TaskList	Array of WorkflowTask	否	工作流任务列表 注意：此字段可能返回 null，表示取不到有效值。
BundleId	String	否	工作流绑定的 Bundle唯一标识，未绑定时为空 注意：此字段可能返回 null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76
BundleInfo	String	否	Bundle信息 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo
GitConfigId	String	否	Git配置ID 注意：此字段可能返回 null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76

名称	类型	必选	描述
GitBranch	String	否	Git分支信息 注意：此字段可能返回 null，表示取不到有效值。 示例值：main

LabelBrief

标签信息

被如下接口引用：CreateWorkflow, GetWorkflow, ListWorkflows, UpdateWorkflow。

名称	类型	必选	描述
LabelKey	String	否	标签名称 注意：此字段可能返回 null，表示取不到有效值。 示例值：key1
LabelValue	String	否	标签值 注意：此字段可能返回 null，表示取不到有效值。 示例值：value1
LabelKeyId	String	否	标签名称ID，可通过标签相关接口获取 注意：此字段可能返回 null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76
LabelValueId	String	否	标签值ID，可通过标签相关接口获取 注意：此字段可能返回 null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76

ListWorkflowsRsp

ListWorkflowsRsp

被如下接口引用：ListWorkflows。

名称	类型	必选	描述
PageNumber	Integer	否	当前页码 注意：此字段可能返回 null，表示取不到有

名称	类型	必选	描述
			效值。 示例值：1
PageSize	Integer	否	每页大小 注意：此字段可能返回 null，表示取不到有效值。 示例值：10
TotalCount	Integer	否	总记录数 注意：此字段可能返回 null，表示取不到有效值。 示例值：1
TotalPageNumber	Integer	否	总页数 注意：此字段可能返回 null，表示取不到有效值。 示例值：1
Items	Array of WorkflowBrief	否	工作流列表 注意：此字段可能返回 null，表示取不到有效值。

MonitorMetricBrief

监控指标配置

被如下接口引用：CreateWorkflow, GetWorkflow, UpdateWorkflow。

名称	类型	必选	描述
MonitorMetricId	String	否	监控指标 ID，创建时无需传入，由服务端生成 注意：此字段可能返回 null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76
AlarmMonitorType	String	否	告警的监控对象类型，如工作流、任务等，当前支持 1. WORKFLOW 2. TASK 注意：此字段可能返回 null，表示取不到有效值。 示例值：WORKFLOW

名称	类型	必选	描述
Metrics	Array of MonitorMetricItem	否	监控指标列表 注意：此字段可能返回 null，表示取不到有效值。

MonitorMetricItem

单个监控指标

被如下接口引用：CreateWorkflow, GetWorkflow, UpdateWorkflow。

名称	类型	必选	描述
MetricType	String	否	监控指标类型,有三种类型：1. RUN_DURATION（运行时长）2. WAIT_DURATION（等待时长）3. COMPLETION_TIME（完成时间） 注意：此字段可能返回 null，表示取不到有效值。 示例值：COMMON
WarningThreshold	String	否	警告阈值，单位为毫秒级别，对于COMPLETION_TIME:从当日时间点00:00起算 注意：此字段可能返回 null，表示取不到有效值。 示例值：1
TimeoutThreshold	String	否	超时阈值，单位为毫秒级别，对于COMPLETION_TIME:从当日时间点00:00起算 注意：此字段可能返回 null，表示取不到有效值。 示例值：1

OrderBy

排序字段

被如下接口引用：ListWorkflows。

名称	类型	必选	描述
Direction	String	否	排序方向，Asc（升序）或 Desc（降序），大小写不敏感 示例值："Desc"
Name	String	否	排序字段名 示例值："CreateTime"

ParamInfo

参数键值对

被如下接口引用：CreateWorkflow, GetWorkflow, UpdateWorkflow。

名称	类型	必选	描述
ParamId	String	否	参数ID，创建时无需传入，由服务端生成 注意：此字段可能返回 null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76
ParamKey	String	否	参数名 注意：此字段可能返回 null，表示取不到有效值。 示例值：key1
ParamValue	String	否	参数值 注意：此字段可能返回 null，表示取不到有效值。 示例值：value1

ResourceGroupInfo

资源组信息

被如下接口引用：ListWorkflows。

名称	类型	必选	描述
ResourceGroupId	String	否	资源组ID 注意：此字段可能返回 null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76
ResourceGroupName	String	否	资源组名称 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo
ResourceGroupStatus	String	否	资源组状态 COMPUTE_RESOURCE_STATUS_UNSPECIFIED 定 COMPUTE_RESOURCE_STATUS_PENDING_CR 待创建 COMPUTE_RESOURCE_STATUS_CREATING 创建 COMPUTE_RESOURCE_STATUS_RUNNING 运行 COMPUTE_RESOURCE_STATUS_STOPPED 已停 COMPUTE_RESOURCE_STATUS_STOPPING 停止 COMPUTE_RESOURCE_STATUS_STARTING 启动

名称	类型	必选	描述
			COMPUTE_RESOURCE_STATUS_UPDATING 更新 COMPUTE_RESOURCE_STATUS_DELETING 删除 COMPUTE_RESOURCE_STATUS_DELETED 已删 COMPUTE_RESOURCE_STATUS_FAILED 失败 注意：此字段可能返回 null，表示取不到有效值。 示例值： COMPUTE_RESOURCE_STATUS_UNSPECIFIED

TaskRetryStrategy

任务重试策略

被如下接口引用：CreateWorkflow, GetWorkflow, ListWorkflows, UpdateWorkflow。

名称	类型	必选	描述
MaxRetryTimes	Integer	否	最多重试次数，默认3 注意：此字段可能返回 null，表示取不到有效值。 示例值：1
RetryBetweenWaitTime	Integer	否	重试之间等待时间，默认5 注意：此字段可能返回 null，表示取不到有效值。 示例值：1
RetryBetweenWaitTimeUnit	String	否	重试之间等待时间单位 毫秒：MILLISECOND秒：SECOND分钟（默认）：MINUTE小时：HOUR 注意：此字段可能返回 null，表示取不到有效值。 示例值：SECOND
TaskRunFailureRetrySwitch	Boolean	否	任务运行失败时重试开关，默认为true 注意：此字段可能返回 null，表示取不到有效值。 示例值：false
TaskRunTimeoutRetrySwitch	Boolean	否	任务运行超时时重试开关，默认为false 注意：此字段可能返回 null，表示取不到有效值。 示例值：false

TaskRunConditionRule

任务运行条件规则

被如下接口引用：CreateWorkflow, GetWorkflow, UpdateWorkflow。

名称	类型	必选	描述
UpstreamTaskId	String	否	上游任务ID 注意：此字段可能返回 null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76
UpstreamTaskName	String	否	上游任务名称 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo
AllowedStates	Array of String	否	任务可运行条件 支持的状态值： - SUCCESS: 成功 - FAILED: 失败 - UPSTREAM_FAILED: 上游失败 - EXCLUDED: 排除运行 注意：此字段可能返回 null，表示取不到有效值。 示例值：["ACTIVE"]

TaskType

`TaskTypePropertyList` 中 `TaskTypeProperty` 针对不同任务类型需要填写不同的 key 和 value；

1. NOTEBOOK 任务类型

属性配置

属性键	属性名称	描述	是否必需
Source	来源	可填2或5,来源 2:GIT, 5:工作空间	是
NotebookPath	Notebook 相对路径	Source为5时，需从（ListFiles）获取	Source 为 2、5 时，必填

2. DATA_INTEGRATION 任务类型

属性配置

属性键	属性名称	描述	是否必需
Source	来源	必填:4,表示来源为COS	是
TemplatePath	数据接入任务配置路径	需从（ListBatchIngestionTasks）接口获取	是
DataIntegrationTaskId	数据接入任务ID	需从（ListBatchIngestionTasks）接口获取	是

3. RUN_WORKFLOW 任务类型

属性配置

属性键	属性名称	描述	是否必需
WorkflowId	选择工作流	需从（ListWorkflows）接口获取	是

4. SQL 任务类型

属性配置

属性键	属性名称	描述	是否必需
Source	来源	可填2或5,来源 2:GIT, 5:工作空间	是
SqlPath	SQL脚本路径	SQL脚本路径	Source 为 2 时，必填
CodeFileName	文件名称	Source为5时，需从（ListReleasedQueries）接口获取	否
CodeFileId	文件ID	Source为5时，需从（ListReleasedQueries）接口获取	Source 为 5 时，必填
CodeFileVersion	文件版本	Source为5时，需从（ListReleasedQueries）接口获取	否

5. PYTHON 任务类型

属性配置

属性键	属性名称	描述	是否必需
Source	来源	可填2或5,来源 2:GIT, 5:工作空间	是
SourcePath	Python脚本路径	Source为5时，需从（ListFiles）接口获取	是

6. DATA_QUALITY（质量监控）任务类型

属性配置

属性键	属性名称	描述	是否必需
Source	来源	必填:4,表示来源为COS	是
TemplatePath	质量监控配置路径	需从（ListDataQualityTaskSummaries）接口获取	是
SourceUniqueId	质量监控ID	需从（ListDataQualityTaskSummaries）接口获取	是
ExecutionType	执行类型	必填:SQL	是
AfterAspect	质量任务后置切面	需从（ListDataQualityTaskSummaries）接口获取	否
BeforeAspect	质量任务前置切面	需从（ListDataQualityTaskSummaries）接口获取	否

7. IF_ELSE 任务类型

属性配置

属性键	属性名称	描述	是否必需
Conditions	条件列表	IF-ELSE条件配置	是

8. FOR_EACH 任务类型

属性配置

属性键	属性名称	描述	是否必需
MaxConcurrency	最大并发数	最大并发数，默认为1	是

属性键	属性名称	描述	是否必需
MaxIterations	最大迭代次数	最大迭代次数，默认1000	是
LoopDataArray	循环参数	JSON格式的数组，或 {{}} 包裹的变量	是

9. RAY_JOB（Ray作业）任务类型

属性配置

属性键	属性名称	描述	是否必需
RunMode	运行方式	运行方式，取值：SERVERLESS（按需拉起集群）/ DEDICATED（提交到指定集群）。默认 DEDICATED	是
Entrypoint	入口指令	入口指令	是
JobConfig	任务配置	如存储配置 + 计算环境。传值请参考前端页面保存Ray作业任务时调用UpdateWorkflow。Image请调用DLC接口ListImages接口Url字段获取	RunMode 为 SERVERLESS 时，必填
ClusterId	Ray 集群	需从所选计算资源下已配置的 Ray 集群（引擎）列表中选择，可通过计算资源相关接口获取	RunMode 为 DEDICATED 时，必填
Source	任务来源	任务来源，取值：1（本地文件）/ 5（工作空间）	是
JobPackage	任务来源文件地址	任务来源路径：Source 为 1 时存本地上传文件的 COS 地址（支持 .zip / .py 文件）；Source 为 5 时存所选工作空间文件/文件夹路径	是
JobPackageName	任务来源本地文件名	任务来源本地文件名	Source 为 1 时，必填
CodeFileId	代码文件ID	Source 为 5 时，需从工作空间文件选择组件获取所选文件/文件夹对应的ID	Source 为 5 时，必填
WorkspaceEntryType	工作空间入口类型	Source 为 5 时，所选工作空间条目的类型：FILE（文件）/ FOLDER（文	Source 为 5 时，必填

属性键	属性名称	描述	是否必需
		件夹），由前端选择器随选择目标自动写入	

RuntimePropertyList

任务运行参数列表，用于配置任务运行时的计算资源，与 TaskTypePropertyList（任务扩展属性）区分：TaskTypePropertyList 承载任务自身的业务配置（如脚本来源、路径等），RuntimePropertyList 承载任务运行时的资源配置（如资源模式、CU规格、Executor数量等）。

主要适用于需要配置计算资源的任务类型（如 NOTEBOOK、PYTHON）。

具体可填写的属性键以任务类型属性配置为准，可通过 ListWorkflowTaskTypeProperties 接口获取（propertyType 为 RUNTIME 的属性项）。常见运行参数键说明：

属性键	属性名称	描述	必填性（联动条件满足时）	默认值
ResourceMode	资源模式	1：分布式；2：单节点	是	1
ConfigType	配置类型	DEFAULT：默认配置； CUSTOM：自定义配置	是	DEFAULT
ExecutorAllocation	Executor分配模式	DYNAMIC：动态分配；FIXED：固定分配	是（分布式且自定义配置时）	DYNAMIC
ExecutorMinNum	Executor最小个数	正整数	是（动态分配时）	1
ExecutorMaxNum	Executor最大个数	正整数	是（动态分配时）	1
ExecutorFixedNum	Executor固定个数	正整数	否（固定分配时）	–
ExecutorCU	Executor资源规格	small / medium / large / xlarge / 4xlarge	否（分布式且自定义配置时）	–
DriverCU	Driver资源规格	small / medium / large / xlarge /	否（分布式且自定义配置时 或者单节点时）	–

属性键	属性名称	描述	必填性（联动条件满足时）	默认值
		4xlarge		
ExecutorGPU	Executor GPU数量	0表示不使用GPU	否（分布式且自定义配置时）	–
DriverGPU	Driver GPU数量	0表示不使用GPU	否（分布式且自定义配置时）	–
Style	配置样式	UI / JSON	是（自定义配置时）	UI

补充说明：

1. "必填性"指属性配置中的必填标记，仅当属性联动条件（如 ResourceMode=1 且 ConfigType=CUSTOM）满足时才触发必填校验；

2. ConfigType 为 DEFAULT（默认配置）时，需调用 ListComputeResourceOptions 接口获取所选计算资源的默认规格值，并将其填充到运行参数（ExecutorCU、DriverCU、ExecutorMinNum、ExecutorMaxNum 等）后传入；ConfigType 为 CUSTOM（自定义配置）时，运行参数由调用方自行指定；

被如下接口引用：CreateWorkflow, GetWorkflow, UpdateWorkflow。

名称	类型	必选	描述
TaskTypeName	String	否	任务类型：SQL：用于执行SQL查询和数据处理操作；DATA_INTEGRATION：用于离线数据接入操作；NOTEBOOK：用于运行Notebook脚本；RUN_WORKFLOW：用于执行嵌套工作流；PYTHON：用于运行Python脚本；RAY_JOB：用于运行Ray作业；DATA_QUALITY：用于数据质量监控；IF_ELSE：用于条件分支判断；FOR_EACH：用于循环遍历执行； 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo
Notebook	TaskTypeNotebookExt	否	Notebook 类型扩展信息 注意：此字段可能返回 null，表示取不

名称	类型	必选	描述
			到有效值。
TaskTypePropertyList	Array of TaskTypeProperty	否	任务扩展属性列表，具体填写参考 ListWorkflowTaskTypePropertie 接口 注意：此字段可能返回 null，表示取不到有效值。
RuntimePropertyList	Array of TaskTypeProperty	否	运行时属性列表 注意：此字段可能返回 null，表示取不到有效值。

TaskTypeNotebookExt

Notebook 类型任务扩展

被如下接口引用：CreateWorkflow, GetWorkflow, UpdateWorkflow。

名称	类型	必选	描述
Source	String	否	脚本来源。取值：SCRIPT_SOURCE_LOCAL（本地）/ SCRIPT_SOURCE_GIT（Git 仓库）/ SCRIPT_SOURCE_CFS（CFS 文件系统）/ SCRIPT_SOURCE_COS（COS 对象存储）/ SCRIPT_SOURCE_WORKSPACE（工作空间） 注意：此字段可能返回 null，表示取不到有效值。 示例值：SCRIPT_SOURCE_LOCAL
DisplayPath	String	否	前端显示使用，对执行平台无意义 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo
NotebookPath	String	否	Notebook 相对路径 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo
NotebookAbsolutePath	String	否	Notebook 绝对路径 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo

TaskTypeProperty

任务类型属性键值对

被如下接口引用：CreateWorkflow, GetWorkflow, UpdateWorkflow。

名称	类型	必选	描述
PropertyKey	String	否	属性名 注意：此字段可能返回 null，表示取不到有效值。 示例值：key1
PropertyValue	String	否	属性值 注意：此字段可能返回 null，表示取不到有效值。 示例值：value1

UpdateWorkflowRsp

UpdateWorkflowRsp

被如下接口引用：UpdateWorkflow。

名称	类型	必选	描述
Status	Boolean	否	更新状态，true 表示成功 注意：此字段可能返回 null，表示取不到有效值。 示例值：false

Workflow

工作流完整配置

被如下接口引用：UpdateWorkflow。

名称	类型	必选	描述
WorkspaceId	String	否	工作空间ID，可通过 ListWorkspaces 获取 注意：此字段可能返回 null，表示取不到有效值。 示例值：17622177773248536
BaseInfo	WorkflowBaseInfo	否	工作流基本信息

名称	类型	必选	描述
			注意：此字段可能返回 null，表示取不到有效值。
Trigger	Array of WorkflowTriggerConfiguration	否	工作流调度配置 注意：此字段可能返回 null，表示取不到有效值。
ParamList	Array of ParamInfo	否	工作流参数列表 参数名必填且只能包含数字、大小写字母、空格、.$@#!%^&*()-_+=><!--'，最长128个字符--> 注意：此字段可能返回 null，表示取不到有效值。
LabelList	Array of LabelBrief	否	标签 标签名必填且只能包含数字、大小写字母、空格、.$@#!%^&*()-_+=><!--'，最长128个字符--> 注意：此字段可能返回 null，表示取不到有效值。
Alarm	AlarmBrief	否	工作流告警配置 注意：此字段可能返回 null，表示取不到有效值。
MonitorMetric	MonitorMetricBrief	否	监控指标配置 注意：此字段可能返回 null，表示取不到有效值。
AdvanceConfig	WorkflowAdvanceConfig	否	工作流高级设置 注意：此字段可能返回 null，表示取不到有效值。
TaskList	Array of WorkflowTask	否	工作流任务列表

名称	类型	必选	描述
			注意：此字段可能返回 null，表示取不到有效值。
BundleId	String	否	BundleId，可通过 Bundle 相关接口获取 注意：此字段可能返回 null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76
BundleInfo	String	否	Bundle信息 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo
GitConfigId	String	否	GIT配置ID，对应GetWorkspaceConfig接口中的ConfigKey 注意：此字段可能返回 null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76
GitBranch	String	否	Git分支信息 注意：此字段可能返回 null，表示取不到有效值。 示例值：main

WorkflowAdvanceConfig

工作流高级设置

被如下接口引用：CreateWorkflow, GetWorkflow, UpdateWorkflow。

名称	类型	必选	描述
QueuingMode	String	否	排队模式，ON（默认）, OFF 注意：此字段可能返回 null，表示取不到有效值。 示例值：OFF
MaxConcurrentNum	Integer	否	默认值为1 QueuingMode为ON时，MaxConcurrentNum 设置才生效；只能输入大于0的整数，输入非法值自动转换为1 注意：此字段可能返回 null，表示取不到有效值。 示例值：1

WorkflowBaseInfo

工作流基本信息（入参用）

被如下接口引用：CreateWorkflow, UpdateWorkflow。

名称	类型	必选	描述
WorkflowName	String	否	工作流名称，长度不超过 1024 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo_workflow
WorkflowId	String	否	工作流ID，创建时无需传入，由服务端生成 注意：此字段可能返回 null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76
RunUserUin	String	否	工作流运行人UIN 注意：此字段可能返回 null，表示取不到有效值。 示例值：100044134096
Description	String	否	描述 注意：此字段可能返回 null，表示取不到有效值。 示例值：描述信息
OwnerUserName	String	否	工作流负责人用户名 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo
CreateUserUin	String	否	创建人UIN。系统生成字段，入参传值不生效（服务端忽略且不报错） 【已废弃】服务端忽略传入值，不报错。

名称	类型	必选	描述
			注意：此字段可能返回 null，表示取不到有效值。 示例值：100044134096

WorkflowBaseInfoDetail

工作流基本信息（出参用，含系统生成字段与负责人展示信息）

被如下接口引用：GetWorkflow。

名称	类型	必选	描述
WorkflowName	String	否	工作流名称 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo_workflow
WorkflowId	String	否	工作流ID 注意：此字段可能返回 null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76
CreateUserUin	String	否	创建人UIN 注意：此字段可能返回 null，表示取不到有效值。 示例值：100044134096
RunUserUin	String	否	工作流运行人UIN 注意：此字段可能返回 null，表示取不到有效值。 示例值：100044134096
Description	String	否	描述 注意：此字段可能返回 null，表示取不到有效值。 示例值：描述信息
OwnerUserName	String	否	工作流负责人用户名 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo
OwnerUserUin	String	否	工作流负责人UIN 注意：此字段可能返回 null，表示取不到有效值。 示例值：100044134096
OwnerDisplayName	String	否	工作流负责人展示名 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo

名称	类型	必选	描述
CreateTime	String	否	创建时间，单位：毫秒时间戳 注意：此字段可能返回 null，表示取不到有效值。 示例值：1773244800000
UpdateTime	String	否	更新时间，单位：毫秒时间戳 注意：此字段可能返回 null，表示取不到有效值。 示例值：1773244800000

WorkflowBrief

工作流列表项

被如下接口引用：ListWorkflows。

名称	类型	必选	描述
WorkflowName	String	否	工作流名称 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo_workflow
WorkflowId	String	否	工作流ID 注意：此字段可能返回 null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76
Description	String	否	描述 注意：此字段可能返回 null，表示取不到有效值。 示例值：描述信息
CreateUserUin	String	否	创建人UIN

名称	类型	必选	描述
			注意：此字段可能返回null，表示取不到有效值。 示例值：100044134096
OwnerUserName	String	否	工作流负责人用户名 注意：此字段可能返回null，表示取不到有效值。 示例值：demo
OwnerUserUin	String	否	工作流负责人UIN 注意：此字段可能返回null，表示取不到有效值。 示例值：100044134096
OwnerDisplayName	String	否	工作流负责人展示名 注意：此字段可能返回null，表示取不到有效值。 示例值：demo
CreateTime	String	否	创建时间，单位：毫秒时间戳 注意：此字段可能返回null，表示取不到有效值。 示例值：1773244800000
UpdateTime	String	否	更新时间，单位：毫秒时间戳 注意：此字段可能返回

名称	类型	必选	描述
			null，表示取不到有效值。 示例值：1773244800000
LabelList	Array of LabelBrief	否	标签列表 注意：此字段可能返回null，表示取不到有效值。
Trigger	Array of WorkflowTriggerConfiguration	否	工作流调度配置 注意：此字段可能返回null，表示取不到有效值。
RunUserUin	String	否	工作流运行人UIN 注意：此字段可能返回null，表示取不到有效值。 示例值：100044134096
RunUserName	String	否	工作流运行人用户名 注意：此字段可能返回null，表示取不到有效值。 示例值：demo
TaskList	Array of WorkflowTaskNodeBrief	否	工作流任务节点列表 注意：此字段可能返回null，表示取不到有效值。
WorkflowRunList	Array of WorkflowRunBrief	否	工作流运行情况列表 注意：此字段可能返回

名称	类型	必选	描述
			null，表示取不到有效值。
ResourceGroupInfoList	Array of ResourceGroupInfo	否	资源组信息列表 注意：此字段可能返回null，表示取不到有效值。
Permission	String	否	工作流权限信息 注意：此字段可能返回null，表示取不到有效值。 示例值：demo
BundleId	String	否	工作流绑定的 Bundle 唯一标识，未绑定时为空 注意：此字段可能返回null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76
BundleInfo	String	否	Bundle信息 注意：此字段可能返回null，表示取不到有效值。 示例值：demo
GitConfigId	String	否	Git配置ID 注意：此字段可能返回null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76

名称	类型	必选	描述
GitBranch	String	否	Git分支信息 注意：此字段可能返回 null，表示取不到有效值。 示例值：main

WorkflowRunBrief

工作流列表项的运行情况

被如下接口引用：ListWorkflows。

名称	类型	必选	描述
WorkflowRunId	String	否	工作流运行ID 注意：此字段可能返回 null，表示取不到有效值。 示例值：8f2a1c40-1d33-4f9a-9c21-6b0a7e5d3e11
RunStartTime	String	否	运行开始时间，单位：毫秒时间戳 注意：此字段可能返回 null，表示取不到有效值。 示例值：1773244800000
RunState	String	否	运行状态 注意：此字段可能返回 null，表示取不到有效值。 示例值：ACTIVE
ErrorCodeString	String	否	运行错误码 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo

WorkflowTask

工作流任务信息。 注意：本结构同时用于入参（CreateWorkflow / UpdateWorkflow）与出参（GetWorkflow）， 其中 CreateTime / UpdateTime / CreateUserUin 为系统生成字段，仅在出参中有值， 入参传值不生效（服务端忽略且不报错）。

被如下接口引用：CreateWorkflow, GetWorkflow, UpdateWorkflow。

名称	类型	必选	描述
ParamList	Array of ParamInfo	否	任务参数 注意：此字段可能返回 null，表示取不到有效值。
DependOnList	Array of DependOnBrief	否	任务依赖 注意：此字段可能返回 null，表示取不到有效值。
TaskId	String	否	任务ID，创建时无需传入，由服务端生成 注意：此字段可能返回 null，表示取不到有效值。 示例值：a4dfb8d2-cc55-4a24-8573-0091b791a927
TaskName	String	否	任务名称 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo
TaskType	TaskType	否	任务类型 注意：此字段可能返回 null，表示取不到有效值。
ResourceGroupId	String	否	资源组ID，可通过资源组相关接口获取 注意：此字段可能返回 null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76
Description	String	否	任务描述 注意：此字段可能返回 null，表示取不到有效值。 示例值：描述信息
Alarm	AlarmBrief	否	任务告警 注意：此字段可能返回 null，表示取不到有效值。
MonitorMetric	MonitorMetricBrief	否	监控指标 注意：此字段可能返回 null，表示取不到有效值。
TaskRetryStrategy	TaskRetryStrategy	否	任务重试策略 注意：此字段可能返回 null，表示取不到有效值。
DependOnRunCondition	String	否	任依赖运行条件 • ALL_SUCCESS: 全部成功：所有上游依赖任务均已执行并成功 • ONE_SUCCESS: 至少一个成功：至少有一个上游依赖任务成功 • NONE_FAILED: 目前没有失败：没有依赖任务失败，并且至少有一个依赖任务在运行中 • ALL_DONE: 全部完成：所有上游依赖任务均已执行并完成（无论成功或失败 • ONE_FAILED: 至少一个失败：至少有一个上游依赖任务失败 • ALL_FAILED: 全部失败：所有上游依赖任务都失败 • ALL_DONE_AT_LEAST_ONE_SUCCESS：上游全部完成至少一个成功: 所有上游依赖任务都达到终态时，进行依赖判断，至少有一个成功，则依赖判断成功，否则就是跳过运行 • ALL_SKIPPED：上游全部完成，没有跳过运行: 所有上游依赖任务都达到终态时，进行依赖判断, 如果上游状态全部都是成功、失败、上游失败状态，则依赖判断成功，否则为跳过运行 • ONE_DONE：至少一个完成：上游只要有一个完成了，就进行依赖判断，且依赖判断成功，否则还是等待上游 • ALL_DONE_NONE_FAILED_AT_LEAST_ONE_SUCCESS：上游全部完成，没有失败，至少有一个成功: 所有上游依赖任务都达到终态时，进行依赖判断，上游没有一个失败且至少有一个成功的情况下，依赖判断成功，否则就是跳过运行 • NONE_SKIPPED：上游全部完成，没有跳过运行: 所有上游依赖任务都达到终态时，进行依赖判断, 如果上游状态全部都是成功、失败、上游失败状态，则依赖判断成功，否则为跳过运行 • ALL_DONE_AT_LEAST_ONE_FAILED：上游全部完成至少一个失败: 所有上游依赖任务都达到终态时，进行依赖判断，至少有一个失败，则依赖判断成功，否则就是跳过运行 • ADVANCED:运行条件为高级模式时配置 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo
LeftCoordinate	Float	否	任务X坐标 注意：此字段可能返回 null，表示取不到有效值。 示例值：50
TopCoordinate	Float	否	任务Y坐标 注意：此字段可能返回 null，表示取不到有效值。 示例值：50
AdvancedDependencyConfig	AdvancedDependencyConfig	否	任务高级运行参数，当DependOnRunCondition为ADVANCED时配置 注意：此字段可能返回 null，表示取不到有效值。
InnerTask	WorkflowTask	否	内嵌任务（FOR_EACH任务的子任务） 注意：此字段可能返回 null，表示取不到有效值。
CreateTime	String	否	创建时间，单位：毫秒时间戳。出参专用，系统生成，入参传值不生效 【已废弃】服务端忽略传入值，不报错。 注意：此字段可能返回 null，表示取不到有效值。 示例值：1773244800000

UpdateTime	String	否	更新时间，单位：毫秒时间戳。出参专用，系统生成，入参传值不生效 【已废弃】服务端忽略传入值，不报错。 注意：此字段可能返回 null，表示取不到有效值。 示例值：1773244800000
CreateUserUin	String	否	创建人UIN。出参专用，系统生成，入参传值不生效 【已废弃】服务端忽略传入值，不报错。 注意：此字段可能返回 null，表示取不到有效值。 示例值：100044134096

WorkflowTaskNodeBrief

工作流列表项中的工作流任务节点简要信息

被如下接口引用：ListWorkflows。

名称	类型	必选	描述
WorkflowId	String	否	工作流ID 注意：此字段可能返回 null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76
TaskId	String	否	任务ID 注意：此字段可能返回 null，表示取不到有效值。 示例值：a4dfb8d2-cc55-4a24-8573-0091b791a927
TaskName	String	否	任务名称 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo
TaskTypeName	String	否	任务类型名称 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo
DependOnList	Array of DependOnBrief	否	任务依赖列表 注意：此字段可能返

名称	类型	必选	描述
			回 null，表示取不到有效值。
ResourceGroupId	String	否	任务资源组ID 注意：此字段可能返回 null，表示取不到有效值。 示例值：f4ec3e79-7368-44c9-8b04-031fad7f1a76
ResourceGroupName	String	否	任务资源组名称 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo
LeftCoordinate	Float	否	任务X坐标 注意：此字段可能返回 null，表示取不到有效值。 示例值：50
TopCoordinate	Float	否	任务Y坐标 注意：此字段可能返回 null，表示取不到有效值。 示例值：50
TaskRetryStrategy	TaskRetryStrategy	否	任务重试策略 注意：此字段可能返回 null，表示取不到有效值。
DependOnRunCondition	String	否	依赖运行条件 注意：此字段可能返回 null，表示取不到有效值。 示例值：demo
AdvancedDependencyConfig	AdvancedDependencyConfig	否	高级依赖配置 注意：此字段可能返回 null，表示取不到有效值。

名称	类型	必选	描述
InnerTask	WorkflowTaskNodeBrief	否	内嵌工作流任务节点 注意：此字段可能返回 null，表示取不到有效值。

WorkflowTriggerAdvancedConfiguration

工作流调度高级配置。

被如下接口引用：CreateWorkflow, GetWorkflow, UpdateWorkflow。

名称	类型	必选	描述
TaskRetryMode	String	否	任务重试模式 注意：此字段可能返回 null，表示取不到有效值。 示例值：COMMON

WorkflowTriggerConfiguration

工作流调度配置。

被如下接口引用：CreateWorkflow, GetWorkflow, ListWorkflows, UpdateWorkflow。

名称	类型	必选	描述
TriggerId	String	否	调度配置ID，创建时 传入，由服务端生成 注意：此字段可能返 null，表示取不到有 示例值：f4ec3e79- 7368-44c9-8b04 031fad7f1a76
SchedulerStatus	String	否	调度状态 启动： START，暂停：PA 注意：此字段可能返 null，表示取不到有 示例值：START
TriggerMode	String	否	触发方式， - 定时触发： TIME_TRIGGER

名称	类型	必选	描述
			- 持续运行：CONTINUE_RUN 注意： - TIME_TRIGGER下，SchedulerStatus、SchedulerTimeZo StartTime、EndTime、ConfigMode、CycleType、CrontabExpressi 填； - CONTINUE_RU 式下，AdvancedC 必填； 注意：此字段可能返回 null，表示取不到有效 示例值：TIME_TRIGGER
SchedulerTimeZone	String	否	调度时区 注意：此字段可能返回 null，表示取不到有效 示例值：Asia/Shan
StartTime	String	否	调度生效时间，单位：时间戳。必须小于EndTime 注意：此字段可能返回 null，表示取不到有效 示例值：1773244800000
EndTime	String	否	调度结束时间，单位：时间戳。必须大于StartTime 注意：此字段可能返回 null，表示取不到有效 示例值：1773244800000

名称	类型	必选	描述
ConfigMode	String	否	配置方式，常规：COMMON，CRON式：CRON_EXPRESS 注意：此字段可能返回 null，表示取不到有效 示例值：COMMON
CycleType	String	否	周期类型：支持的类型 ONEOFF_CYCLE 一次性 YEAR_CYCL MONTH_CYCLE: WEEK_CYCLE: 周 DAY_CYCLE: 天 HOUR_CYCLE: 小 MINUTE_CYCLE: CRONTAB_CYCL crontab表达式类型 注意：此字段可能返回 null，表示取不到有效 示例值：ONEOFF_CYCLE
CrontabExpression	String	否	cron表达式 注意：此字段可能返回 null，表示取不到有效 示例值：0 0 0 * * ?
ExtraInfo	String	否	Json格式，对账使用 注意：此字段可能返回 null，表示取不到有效 示例值：demo
AdvancedConfig	WorkflowTriggerAdvancedConfiguration	否	高级配置 注意：此字段可能返回 null，表示取不到有效

错误码

最近更新时间：2026-09-11 14:18:58

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 s
ignature 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。

错误码	说明
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。

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

业务错误码

错误码	说明
AuthFailure	CAM签名/鉴权错误。
FailedOperation.CallThirdPartApiError	调用 <serviceName> 服务的接口 <apiName> 失败：<message>。
FailedOperation.CreateWorkflowFailed	FailedOperation.CreateWorkflowFailed
FailedOperation.LabelCountLimit	标签数量已达到最大允许限制
FailedOperation.UpdateWorkflowFailed	FailedOperation.UpdateWorkflowFailed
FailedOperation.WorkflowBundleNoPermission	FailedOperation.WorkflowBundleNoPermission
FailedOperation.WorkflowCountLimit	工作流数量超过10000上限
FailedOperation.WorkflowCreateLockAcquireFailed	获取工作流名称分布式锁失败
FailedOperation.WorkflowNoPermission	无操作权限
InvalidParameterValue.DuplicateTaskNameError	同一工作流内存在重名任务
InvalidParameterValue.ListWorkflowFilterParamError	InvalidParameterValue.ListWorkflowFilterParamError
InvalidParameterValue.LoopDataArrayElementCountLimit	InvalidParameterValue.LoopDataArrayElementCountLimit
InvalidParameterValue.LoopDataArrayJsonInvalid	InvalidParameterValue.LoopDataArrayJsonInvalid
InvalidParameterValue.LoopDataArrayNotJsonArray	InvalidParameterValue.LoopDataArrayNotJsonArray
InvalidParameterValue.LoopDataArrayNotJsonArrayLiteral	InvalidParameterValue.LoopDataArrayNotJsonArrayLiteral
InvalidParameterValue.LoopDataArrayValueBlank	InvalidParameterValue.LoopDataArrayValueBlank
InvalidParameterValue.LoopDataArrayValueLengthLimit	InvalidParameterValue.LoopDataArrayValueLengthLimit
InvalidParameterValue.LoopDataArrayVariableExpressionInvalid	InvalidParameterValue.LoopDataArrayVariableExpressionInvalid
InvalidParameterValue.ParamBlankError	参数 <parameter> 不能为空。
InvalidParameterValue.ParamIllegalError	参数 <parameter> 不符合要求：<message>
InvalidParameterValue.TaskHookValidationFailed	引用对象不存在或已被删除
InvalidParameterValue.TaskNameContainsIllegalCharactersError	任务名包含非法字符
InvalidParameterValue.TaskNameExceedsLimitError	任务名超过128字符限制
InvalidParameterValue.WorkflowNameExists	InvalidParameterValue.WorkflowNameExists
InvalidParameterValue.WorkflowNameInvalid	InvalidParameterValue.WorkflowNameInvalid
InvalidParameterValue.WorkflowStartTimeAfterEndTimeError	工作流调度开始时间不能晚于结束时间
ResourceNotFound.OneFlowResourceNoExistError	资源不存在或已被删除
ResourceNotFound.WorkflowNotExist	ResourceNotFound.WorkflowNotExist
ResourceNotFound.WorkflowNotFound	ResourceNotFound.WorkflowNotFound
ResourceNotFound.WorkflowTriggerNotFound	ResourceNotFound.WorkflowTriggerNotFound