云托付物理服务器

API 文档

腾讯云

文档目录

API 文档
更新历史

最近更新时间：2025-06-06 01:18:20

第 7 次发布

发布时间：2025-06-06 01:18:04

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- ExportCustomerWorkOrderDetail

第 6 次发布

发布时间：2025-05-27 01:17:29

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- DescribeRacks
  - 新增入参：RackName

修改数据结构：

- CustomerReceipt
  - 新增成员：IDCardType
- SelfOperation
  - 新增成员：IDCardType

第 5 次发布

发布时间：2025-05-08 01:17:25

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- CreateReceivingWorkOrder
  - 新增入参：WithRackOn, DeviceRackOnList, StuffOption, SelfOperationInfo, WithPowerOn

第 4 次发布

发布时间：2025-04-29 01:17:57

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- OrderStep
  - 新增成员：OwnerPhone

第 3 次发布

发布时间：2025-04-21 01:11:34

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- DescribeModelVersionList
  - 新增入参：ModelName

第 2 次发布

发布时间：2025-04-11 01:17:29

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- CreateSpeciallyQuitWorkOrder

第 1 次发布

发布时间：2025-03-24 09:42:38

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- ConfirmCommonServiceWorkOrder
- CreateCommonServiceWorkOrder
- CreateModelEvaluationWorkOrder
- CreateMovingWorkOrder
- CreateNetDeviceModel
- CreatePersonnelVisitWorkOrder
- CreatePowerOffWorkOrder
- CreatePowerOnWorkOrder
- CreateQuitWorkOrder
- CreateRackOffWorkOrder
- CreateRackOnWorkOrder
- CreateReceivingWorkOrder
- CreateServerModel
- DescribeAvailableModelList
- DescribeCampusList
- DescribeCommonServiceWorkOrderDetail
- DescribeCustomerInfo
- DescribeDeviceList
- DescribeDeviceWorkOrderDetail
- DescribeIdcUnitAssetDetail

- DescribeIdcUnitDetail
- DescribeIdcs
- DescribeModel
- DescribeModelEvaluationWorkOrderDetail
- DescribeModelTemplate
- DescribeModelVersionList
- DescribePersonnelVisitWorkOrderDetail
- DescribePositionStatusSummary
- DescribePositions
- DescribeRacks
- DescribeRacksDistribution
- DescribeResourceUsage
- DescribeWorkOrderList
- DescribeWorkOrderStatistics
- DescribeWorkOrderTypes
- ModifyWorkOrderTypeCollectFlag

新增数据结构：

- AvailableModelVersion
- Cage
- Campus
- CommonServiceBaseInfo
- CustomerInfo
- CustomerReceipt
- Device
- DeviceHistory
- DeviceOrderBaseInfo
- DevicePosition
- DeviceRackOn
- Distribution
- ExpressDelivery
- Filter
- Idc
- IdcUnit
- IdcUnitInfo
- LogisticsReceipt
- ModelEvaluationBaseInfo
- ModelVersion
- ModelVersionCount
- ModelVersionDetail
- NetDeviceModel
- NetReceivingInfo
- OptionValueItem
- OrderStep
- OtherDevReceivingInfo
- Personnel
- PersonnelVisitBaseInfo
- Position
- PositionStatusItem
- PowerOffConfirm

- Rack
- RackUsage
- SelfOperation
- ServerModel
- ServerReceivingInfo
- TemplateOption
- WireReceivingInfo
- WorkOrderData
- WorkOrderFamilyDetail
- WorkOrderTinyInfo
- WorkOrderTypeDetail

简介

最近更新时间：2025-04-25 01:16:42

云托付物理服务器（CHC）是腾讯云全新推出的托管型基础设施类云产品，腾讯提前完成网络机位建设，客户仅需自服务器并托管在腾讯云机房内，并可以共用云上VPC能力，可快速无缝对接云上资源

API 概览

最近更新时间：2025-06-06 01:18:20

资源相关接口

接口名称	接口功能	频率限制（次/秒）
DescribeCampusList	获取园区列表	20
DescribeCustomerInfo	查询客户信息	20
DescribeDeviceList	获取设备列表	20
DescribeIdcs	获取机房和机房管理单元信息	20
DescribePositions	获取机位列表	20
DescribeRacks	获取机架列表	20
DescribeRacksDistribution	获取机位分布	20
DescribeResourceUsage	查询资源汇总	20

工单相关接口

接口名称	接口功能	频率限制（次/秒）
ConfirmCommonServiceWorkOrder	确认通用服务工单	20
CreateCommonServiceWorkOrder	创建通用服务工单	20
CreateModelEvaluationWorkOrder	创建设备型号评估工单	20
CreateMovingWorkOrder	创建设备搬迁工单	20
CreateNetDeviceModel	创建网络设备新型号	20
CreatePersonnelVisitWorkOrder	创建人员到访工单	20
CreatePowerOffWorkOrder	创建设备关电工单	20
CreatePowerOnWorkOrder	创建设备开电工单	20
CreateQuitWorkOrder	创建设备退出工单	20
CreateRackOffWorkOrder	创建设备下架工单	20
CreateRackOnWorkOrder	创建设备上架工单	20
CreateReceivingWorkOrder	创建设备收货工单	20
CreateServerModel	创建服务器型号	20
CreateSpeciallyQuitWorkOrder	创建临时设备退出工单	20
DescribeAvailableModelList	获取机房内可用的型号列表	20
DescribeCommonServiceWorkOrderDetail	获取通用服务工单详情	20
DescribeDeviceWorkOrderDetail	查询设备类工单详情	20

接口名称	接口功能	频率限制（次/秒）
DescribeIdcUnitAssetDetail	查询机房管理单元资产详情	20
DescribeIdcUnitDetail	查询机房管理单元详情	20
DescribeModel	查询设备型号详情	20
DescribeModelEvaluationWorkOrderDetail	查询设备型号评估工单	20
DescribeModelTemplate	获取型号的填写模板	20
DescribeModelVersionList	获取型号和版本列表	20
DescribePersonnelVisitWorkOrderDetail	查询人员到访工单详情	20
DescribePositionStatusSummary	查询机位状态的汇总数据	20
DescribeWorkOrderList	查询工单列表	20
DescribeWorkOrderStatistics	查询工单统计信息	20
DescribeWorkOrderTypes	查询用户可用的工单列表	20
ExportCustomerWorkOrderDetail	导出客户工单详情	20
ModifyWorkOrderTypeCollectFlag	修改工单类型的收藏标识	20

⚠ 注意：

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

调用方式
请求结构

最近更新时间：2025-05-21 01:12:07

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

2. 通信协议

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

3. 请求方法

支持的 HTTP 请求方法：

- POST（推荐）
- GET

POST 请求支持的 Content-Type 类型：

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

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

4. 字符编码

均使用 `UTF-8` 编码。

公共参数

最近更新时间：2025-03-24 09:43:14

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 chc；tc3_request 为固定字符串； - SignedHeaders：参与签名计算的头部信息，content-type 和 host 为必选头部； - Signature：签名摘要，计算过程详见 文档。
Token	String	否	HTTP 请求头：X-TC-Token。即 安全凭证服务 所颁发的临时安全凭证中的 Token，使用时需要将 SecretId 和 SecretKey 的值替换为临时安全凭证中的 TmpSecretId 和 TmpSecretKey。使用长期密钥时不能设置此 Token 字段。
Language	String	否	HTTP 请求头：X-TC-Language。指定接口返回的语言，仅部分接口支持此参数。取值：zh-CN，en-US。zh-CN 返回中文，en-US 返回英文。

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

HTTP GET 请求结构示例：

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

Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2018-10-09/cvm/tc3_request,
SignedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
Content-Type: application/x-www-form-urlencoded
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1539084154
X-TC-Region: ap-guangzhou
```

HTTP POST（application/json）请求结构示例：

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

Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2018-05-30/cvm/tc3_request,
SignedHeaders=content-type;host, Signature=582c400e06b5924a6f2b5d7d672d79c15b13162d9279b0855cfba6789a8edb4c
Content-Type: application/json
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1527672334
X-TC-Region: ap-guangzhou

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

HTTP POST（multipart/form-data）请求结构示例（仅特定的接口支持）：

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

Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2018-05-30/cvm/tc3_request,
SignedHeaders=content-type;host, Signature=582c400e06b5924a6f2b5d7d672d79c15b13162d9279b0855cfba6789a8edb4c
Content-Type: multipart/form-data; boundary=58731222010402
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1527672334
X-TC-Region: ap-guangzhou

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

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

10
--58731222010402--
```

使用签名方法 v1 的公共参数

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

参数名称	类型	必选	描述
Action	String	是	操作的接口名称。取值参考接口文档中输入参数章节关于公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。
Region	String	-	地域参数，用来标识希望操作哪个地域的数据。接口接受的地域取值参考接口文档中输入参数公共参数 Region 的说明。**注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。**
Timestamp	Integer	是	当前 UNIX 时间戳，可记录发起 API 请求的时间。例如1529223702，如果与当前时间相差过大，会引起签名过期错误。
Nonce	Integer	是	随机正整数，与 Timestamp 联合起来，用于防止重放攻击。
SecretId	String	是	在 云API密钥 上申请的标识身份的 SecretId，一个 SecretId 对应唯一的 SecretKey ，而 SecretKey 会用来生成请求签名 Signature。
Signature	String	是	请求签名，用来验证此次请求的合法性，需要用户根据实际的输入参数计算得出。具体计算方法参见文档。
Version	String	是	操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
SignatureMethod	String	否	签名方式，目前支持 HmacSHA256 和 HmacSHA1。只有指定此参数为 HmacSHA256 时，才使用 HmacSHA256 算法验证签名，其他情况均使用 HmacSHA1 验证签名。
Token	String	否	即 安全凭证服务 所颁发的临时安全凭证中的 Token，使用时需要将 SecretId 和 SecretKey 的值替换为临时安全凭证中的 TmpSecretId 和 TmpSecretKey。使用长期密钥时不能设置此 Token 字段。
Language	String	否	指定接口返回的语言，仅部分接口支持此参数。取值：zh-CN，en-US。zh-CN 返回中文，en-US 返回英文。

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

HTTP GET 请求结构示例:

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

Host: cvm.tencentcloudapi.com
```

HTTP POST 请求结构示例:

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

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

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

地域列表

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

地域	取值
华南地区（广州）	ap-guangzhou

签名方法 v3

最近更新时间：2025-03-24 09:43:15

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

推荐使用 API Explorer　　点击调试

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

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

为什么要进行签名

签名通过以下方式帮助保护请求：

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

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

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

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

申请安全凭证

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

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

申请安全凭证的具体步骤如下：

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

签名版本 v3 签名过程

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

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

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

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

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

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

下面详细解释签名计算过程。

1. 拼接规范请求串

按如下伪代码格式拼接规范请求串（CanonicalRequest）：

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

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

字段名称	解释
	2. 多个头部，按照头部 key（小写）的 ASCII 升序进行拼接。 此示例计算结果是 `content-type:application/json; charset=utf-8\nhost:cvm.tencentcloudapi.com\nx-tc-action:describeinstances\n`。 注意：content-type 必须和实际发送的相符合，有些编程语言网络库即使未指定也会自动添加 charset 值，如果签名时和发送时不一致，服务器会返回签名校验失败。
SignedHeaders	参与签名的头部信息，说明此次请求有哪些头部参与了签名，和 CanonicalHeaders 包含的头部内容是一一对应的。content-type 和 host 为必选头部。 拼接规则： 1. 头部 key 统一转成小写； 2. 多个头部 key（小写）按照 ASCII 升序进行拼接，并且以分号（;）分隔。 此示例为 content-type;host;x-tc-action
HashedRequestPayload	请求正文（payload，即 body，此示例为 `{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}`）的哈希值，计算伪代码为 Lowercase(HexEncode(Hash.SHA256(RequestPayload)))，即对 HTTP 请求正文做 SHA256 哈希，然后十六进制编码，最后编码串转换成小写字母。对于 GET 请求，RequestPayload 固定为空字符串。此示例计算结果是 `35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064`。

根据以上规则，示例中得到的规范请求串如下：

```
POST
/

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

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

2. 拼接待签名字符串

按如下格式拼接待签名字符串：

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

字段名称	解释
Algorithm	签名算法，目前固定为 `TC3-HMAC-SHA256`。
RequestTimestamp	请求时间戳，即请求头部的公共参数 X-TC-Timestamp 取值，取当前时间 UNIX 时间戳，精确到秒。此示例取值为 `1551113065`。
CredentialScope	凭证范围，格式为 Date/service/tc3_request，包含日期、所请求的服务和终止字符串（tc3_request）。Date 为 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的

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

注意：

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

根据以上规则，示例中得到的待签名字符串如下：

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

3. 计算签名

1）计算派生签名密钥，伪代码如下：

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

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

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

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

2）计算签名，伪代码如下：

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

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

4. 拼接 Authorization

按如下格式拼接 Authorization：

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

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

根据以上规则，示例中得到的值为：

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

最终完整的调用信息如下：

```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2019-02-25/cvm/tc3_request,
SignedHeaders=content-type;host;x-tc-action, Signature=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b4
6093f6ab6c4f
Content-Type: application/json; charset=utf-8
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1551113065
X-TC-Region: ap-guangzhou

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

> ⚠ **注意：**
>
> **请求发送时的 HTTP 头部（Header）和请求体（Payload）必须和签名计算过程中的内容完全一致，否则会返回签名不一致错误。可以通过**

> 打印实际请求内容，网络抓包等方式对比排查。

签名演示

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

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

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

- Signature Demo

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

Java

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

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

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

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

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

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

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

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

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

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

```
String authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + credentialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
System.out.println(authorization);

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

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

Python

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

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

service = "cvm"
host = "cvm.tencentcloudapi.com"
endpoint = "https://" + host
region = "ap-guangzhou"
action = "DescribeInstances"
version = "2017-03-12"
algorithm = "TC3-HMAC-SHA256"
#timestamp = int(time.time())
timestamp = 1551113065
date = datetime.utcfromtimestamp(timestamp).strftime("%Y-%m-%d")
```

```
params = {"Limit": 1, "Filters": [{"Values": [u"未命名"], "Name": "instance-name"}]}

# ************* 步骤 1：拼接规范请求串 *************
http_request_method = "POST"
canonical_uri = "/"
canonical_querystring = ""
ct = "application/json; charset=utf-8"
payload = json.dumps(params)
canonical_headers = "content-type:%s\nhost:%s\nx-tc-action:%s\n" % (ct, host, action.lower())
signed_headers = "content-type;host;x-tc-action"
hashed_request_payload = hashlib.sha256(payload.encode("utf-8")).hexdigest()
canonical_request = (http_request_method + "\n" +
canonical_uri + "\n" +
canonical_querystring + "\n" +
canonical_headers + "\n" +
signed_headers + "\n" +
hashed_request_payload)
print(canonical_request)

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


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

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

print('curl -X POST ' + endpoint
+ ' -H "Authorization: ' + authorization + '"'
+ ' -H "Content-Type: application/json; charset=utf-8"'
+ ' -H "Host: ' + host + '"'
```

```
+ ' -H "X-TC-Action: ' + action + '"'
+ ' -H "X-TC-Timestamp: ' + str(timestamp) + '"'
+ ' -H "X-TC-Version: ' + version + '"'
+ ' -H "X-TC-Region: ' + region + '"'
+ " -d '" + payload + "'")
```

Golang

```
package main

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

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

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

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

// step 1: build canonical request string
httpRequestMethod := "POST"
canonicalURI := "/"
canonicalQueryString := ""
```

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

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

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

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

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

```
fmt.Println(curl)
}
```

PHP

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

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

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

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

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

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

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

Ruby

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

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

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

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

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

puts canonical_request

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

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

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

```
Signature=#{signature}"
puts authorization

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

DotNet

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

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

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

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

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

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

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

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

Dictionary<string, string> headers = new Dictionary<string, string>();
headers.Add("Authorization", authorization);
headers.Add("Host", endpoint);
```

```
headers.Add("Content-Type", contentType + "; charset=utf-8");
headers.Add("X-TC-Timestamp", requestTimestamp.ToString());
headers.Add("X-TC-Version", version);
headers.Add("X-TC-Action", action);
headers.Add("X-TC-Region", region);
return headers;
}
public static void Main(string[] args)
{
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID*******************************
string SECRET_ID = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
string SECRET_KEY = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_KEY");

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

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

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

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

NodeJS

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

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

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

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

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

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

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

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

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

```
console.log(canonicalRequest)

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

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

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

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

C++

```
#include <algorithm>
#include <cstdlib>
#include <iostream>
#include <iomanip>
#include <sstream>
#include <string>
#include <stdio.h>
#include <time.h>
```

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

using namespace std;

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

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

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

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

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

```
#else
h = HMAC_CTX_new();
#endif

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

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

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

return (ss.str());
}

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

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

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

string service = "cvm";
```

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

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

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

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

// ************* 步骤 4：拼接 Authorization *************
string authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + credentialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
cout << authorization << endl;
```

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

C

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

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

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

void hmac_sha256(const char* key, int key_len,
const char* input, int input_len,
```

```
unsigned char* output, unsigned int* output_len)
{
HMAC_CTX *h;
#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX hmac;
HMAC_CTX_init(&hmac);
h = &hmac;
#else
h = HMAC_CTX_new();
#endif

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


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

}

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

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

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

int main()
```

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

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

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


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

```
credential_scope, hashed_canonical_request);
printf("%s\n", string_to_sign);

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

char signature[128] = {0};
hex_encode(k_hmac_sha_sign, output_len, signature);
printf("%s\n", signature);

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

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

其他语言

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

签名失败

存在以下签名失败的错误码，请根据实际情况处理。

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

签名方法

最近更新时间：2025-03-24 09:43:15

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

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

推荐使用 API Explorer　　点击调试

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

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

1. 申请安全凭证

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

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

申请安全凭证的具体步骤如下：

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

注意：每个账号最多可以拥有两对 SecretId/SecretKey。

2. 生成签名串

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

假设用户的 SecretId 和 SecretKey 分别是：

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

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

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

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

参数名称	中文	参数值
Region	实例所在区域	ap-guangzhou
InstanceIds.0	待查询的实例 ID	ins-09dx96dg
Offset	偏移量	0
Limit	最大允许输出	20
Version	接口版本号	2017-03-12

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

2.1. 对参数排序

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

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

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

2.2. 拼接请求字符串

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

然后将格式化后的各个参数用"&"拼接在一起，最终生成的请求字符串为:

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

2.3. 拼接签名原文字符串

此步骤生成签名原文字符串。
签名原文字符串由以下几个参数构成:

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

4. 请求字符串: 即上一步生成的请求字符串。

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

示例的拼接结果为：

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

2.4. 生成签名串

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

具体代码如下，以 PHP 语言为例：

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

最终得到的签名串为：

```
7RAM2xfNMO9EiVTNmPg06MRnCvQ=
```

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

3. 签名串编码

生成的签名串并不能直接作为请求参数，需要对其进行 URL 编码。

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

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

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

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

4. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理。

错误代码	错误描述
AuthFailure.SignatureExpire	签名过期

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

5. 签名演示

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

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

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

- Signature Demo

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

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

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

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

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

Java

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

public class TencentCloudAPIDemo {
```

```
private final static String CHARSET = "UTF-8";

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

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

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

public static void main(String[] args) throws Exception {
TreeMap<String, Object> params = new TreeMap<String, Object>(); // TreeMap可以自动排序
// 实际调用时应当使用随机数，例如: params.put("Nonce", new Random().nextInt(java.lang.Integer.MAX_VALUE));
params.put("Nonce", 11886); // 公共参数
// 实际调用时应当使用系统当前时间，例如: params.put("Timestamp", System.currentTimeMillis() / 1000);
params.put("Timestamp", 1465185768); // 公共参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
params.put("SecretId", System.getenv("TENCENTCLOUD_SECRET_ID")); // 公共参数
params.put("Action", "DescribeInstances"); // 公共参数
params.put("Version", "2017-03-12"); // 公共参数
params.put("Region", "ap-guangzhou"); // 公共参数
params.put("Limit", 20); // 业务参数
params.put("Offset", 0); // 业务参数
params.put("InstanceIds.0", "ins-09dx96dg"); // 业务参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
params.put("Signature", sign(getStringToSign(params), System.getenv("TENCENTCLOUD_SECRET_KEY"), "HmacSHA
1")); // 公共参数
System.out.println(getUrl(params));
}
}
```

Python

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

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

import requests

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

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

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

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

Golang

```
package main

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

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

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

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

for i := range keys {
k := keys[i]
buf.WriteString(k)
buf.WriteString("=")
buf.WriteString(params[k])
buf.WriteString("&")
```

```
}
buf.Truncate(buf.Len() - 1)

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

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

PHP

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

ksort($param);

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

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

Ruby

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

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

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

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

DotNet

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

```
public class Application {
public static string Sign(string signKey, string secret)
{
string signRet = string.Empty;
using (HMACSHA1 mac = new HMACSHA1(Encoding.UTF8.GetBytes(signKey)))
{
byte[] hash = mac.ComputeHash(Encoding.UTF8.GetBytes(secret));
signRet = Convert.ToBase64String(hash);
}
return signRet;
}
public static string MakeSignPlainText(SortedDictionary<string, string> requestParams, string requestMetho
d, string requestHost, string requestPath)
{
string retStr = "";
retStr += requestMethod;
retStr += requestHost;
retStr += requestPath;
retStr += "?";
string v = "";
foreach (string key in requestParams.Keys)
{
v += string.Format("{0}={1}&", key, requestParams[key]);
}
retStr += v.TrimEnd('&');
return retStr;
}

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

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

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

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

NodeJS

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

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

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

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

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

```
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
const SECRET_KEY = process.env.TENCENTCLOUD_SECRET_KEY

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

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

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

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

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

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

返回结果

最近更新时间：2025-03-24 09:43:15

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

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

正确返回结果

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

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

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

错误返回结果

若调用失败，其返回值示例如下为：

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

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

公共错误码

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

参数类型

最近更新时间：2025-03-24 09:43:15

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。

工单相关接口
查询设备型号详情

最近更新时间：2025-04-25 01:16:39

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

查询设备型号详情

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeModel。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
DevModel	是	String	服务器设备型号 示例值：SFT1502-III
CampusId	是	Integer	园区ID 示例值：1
DeviceType	是	String	设备类型，服务器传入 server，网络设备传入 netDevice 示例值：server
Checked	否	Boolean	是否只返回已评估的版本 示例值：false

3. 输出参数

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

4. 示例

示例1 查看服务器设备型号详情

输入示例

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

{
"DevModel": "IBM X3650 M5007",
"CampusId": 163,
"DeviceType": "server"
}
```

输出示例

```
{
"Response": {
"ModelSet": [
{
"CheckResult": 0,
"OptionSet": [
{
"CompareType": "--",
"InputHint": "",
"InputInfo": "请注意节点数代表服务器子机数量，正常情况下标准服务器均为1",
"InputType": "dropDown",
"OptionKey": "devNode",
"OptionName": "节点数",
"OptionValue": "1",
"Standard": "--",
"StandardInfo": "--",
"ValueType": "number"
},
{
"CompareType": "--",
"InputHint": "填写设备高度（正整数，不带U）",
"InputInfo": "",
"InputType": "int",
"OptionKey": "devHeight",
"OptionName": "高度(u)",
"OptionValue": "22",
"Standard": "1u=44.45mm",
"StandardInfo": "1u=44.45mm",
"ValueType": "number"
},
{
"CompareType": "<",
"InputHint": "填写正整数",
```

```
"InputInfo": "",
"InputType": "int",
"OptionKey": "powerEnergy",
"OptionName": "功耗(W)",
"OptionValue": "699",
"Standard": "4400",
"StandardInfo": "单柜<4400/6600",
"ValueType": "number"
}
],
"Version": "V1"
}
],
"RequestId": "c1957d1a-3d98-4463-83ee-d12896ab5bde"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

查询机房管理单元详情

最近更新时间：2025-04-25 01:16:39

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

查询机房管理单元详情

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeIdcUnitDetail。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
IdcUnitId	否	Integer	机房管理单元ID 示例值：12417

3. 输出参数

参数名称	类型	描述
IdcUnitDetail	IdcUnitInfo	机房管理单元详情 示例值：{ "Address": "山东省济南市经十西路宋庄立交南首担山屯中国联通国家数据中心担山屯南楼二楼南机房", "Operator": "tonyzhu", "TelNumber": "0531-87581111" }
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 查询机房管理单元信息

输入示例

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

```
{
"IdcUnitId": 2563
}
```

输出示例

```
{
"Response": {
"IdcUnitDetail": {
"Address": "天津市滨海新区第六大街与北海路口泰达服务外包产业园信环南街9号",
"Operator": "zhangsan",
"TelNumber": ""
},
"RequestId": "545f1583-698c-4d9a-92fe-544f100769c2"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

查询机房管理单元资产详情

最近更新时间：2025-04-25 01:16:39

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

查询机房管理单元资产详情

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeIdcUnitAssetDetail。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
IdcUnitId	否	Integer	机房管理单元ID 示例值：12417

3. 输出参数

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

4. 示例

示例1 查询机房管理单元信息

输入示例

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

{
```

```
"IdcUnitId": 2569
}
```

输出示例

```
{
"Response": {
"IdcUnitDetail": {
"Address": "天津市滨海新区第六大街与北海路口泰达服务外包产业园信环南街",
"AssetManager": "张三",
"AssetManagerTelNumber": "1380013800",
"Operator": "zhangsan",
"TelNumber": ""
},
"RequestId": "8616a5eb-6edd-4d76-b081-fc4c429c3162"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

查询设备类工单详情

最近更新时间：2025-04-25 01:16:39

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

用于查询设备类工单的工单详情，如：'receiving', 'rackOn', 'powerOn', 'powerOff', 'rackOff', 'quit'

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：DescribeDeviceWorkOrderDetail。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
OrderId	是	String	工单ID 示例值：ord-24121717264943561

3. 输出参数

参数名称	类型	描述
OrderId	String	工单ID 示例值：ord-25032218485126792
ServiceType	String	服务类型 示例值：rackOn
OrderType	String	工单类型 示例值：powerOn
OrderStatus	String	工单状态 示例值：finish
StepSet	Array of OrderStep	工单流程状态
DeviceSet	Array of DeviceHistory	工单设备信息
BaseInfo	DeviceOrderBaseInfo	工单的入参信息
RejectReason	String	工单的拒绝原因，工单状态为reject的时候返回 示例值：拒绝原因

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

4. 示例

示例1 查询设备类工单详情

输入示例

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

{
"OrderId": "ord-25022518110325063"
}
```

输出示例

```
{
"Response": {
"BaseInfo": {
"DeviceType": "server",
"IdcId": 373,
"IdcName": "天津数据备份中心东区3栋DC",
"IsPowerOffConfirm": "1",
"Remark": "",
"StuffOption": "2"
},
"DeviceSet": [
{
"AssetId": "TH2502250006E",
"DeviceType": "server",
"IdcId": 373,
"IdcName": "天津数据备份中心东区3栋DC",
"IdcUnitId": 2559,
"IdcUnitName": "天津数据备份中心东区3栋DCM204",
"PositionCode": 4,
"RackName": "M204-J10",
"Sn": "lihwli10225011"
},
{
"AssetId": "TH2502250006D",
"DeviceType": "server",
"IdcId": 373,
"IdcName": "天津数据备份中心东区3栋DC",
```

```
"IdcUnitId": 2559,
"IdcUnitName": "天津数据备份中心东区3栋DCM204",
"PositionCode": 3,
"RackName": "M204-J10",
"Sn": "lihwli10225012"
}
],
"OrderId": "ord-25022518110325063",
"OrderStatus": "finish",
"OrderType": "rackOff",
"RequestId": "c03f33b4-e3e3-4c22-9af3-e8791cee35df",
"ServiceType": "rackOff",
"StepSet": [
{
"FinishTime": "2025-02-25 18:11:03",
"OwnerName": "godwin2@test.com",
"StepName": "发起申请",
"StepStatus": "finish"
},
{
"FinishTime": "2025-02-25 18:12:05",
"OwnerName": "",
"StepName": "数经审核",
"StepStatus": "finish"
},
{
"FinishTime": "2025-02-25 18:18:07",
"OwnerName": "",
"StepName": "现场实施",
"StepStatus": "finish"
}
]
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

导出客户工单详情

最近更新时间：2025-06-09 01:17:42

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

导出工单详情

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：ExportCustomerWorkOrderDetail。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
WorkOrderType.N	是	Array of String	服务工单类型 示例值：['rackPowerOn']
BeginDateTime	是	String	要导出的工单的起始时间 示例值：2024-11-22 00:00:00
EndDateTime	是	String	要导出的工单的结束时间 示例值：2025-05-11 00:00:00

3. 输出参数

参数名称	类型	描述
DownloadUrl	String	返回下载地址
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 导出客户工单详情

输入示例

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

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

{
"WorkOrderType": [
"rackOn"
],
"BeginDateTime": "2025-04-01 00:00:00",
"EndDateTime": "2025-04-30 00:00:00"
}
```

输出示例

```
{
"Response": {
"DownloadUrl": "http://ap-guangzhou-test-1305532550.cos.ap-guangzhou.myqcloud.com/idc_portal_work_order/251233270/251233270_2025-04-01_00-00-00_2025-04-30_00-00-00.xlsx?q-sign-algorithm=sha1&q-ak=xxx&q-sign-time=1749127181%3B1749127541&q-key-time=1749127181%3B1749127541&q-header-list=host&q-url-param-list=&q-signature=3a30d015e08fb524f75b31217d2436bbce70e6ec",
"RequestId": "591f555e-23b9-4b9a-aab8-186364534732"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameterValue.EndTimeLowerThanStartTime	结束时间必须大于起始时间
InvalidParameterValue.InvalidTimeFormat	时间格式不符合规范
InvalidParameterValue.InvalidWorkOrderType	非法的工单类型

获取通用服务工单详情

最近更新时间：2025-04-25 01:16:39

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

查询通用服务工单详情

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeCommonServiceWorkOrderDetail。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
OrderId	是	String	工单ID 示例值：ord-24120215310024323

3. 输出参数

参数名称	类型	描述
StepSet	Array of OrderStep	进度
BaseInfo	CommonServiceBaseInfo	基本信息
DeviceSet	Array of DevicePosition	设备信息
OrderStatus	String	工单状态 示例值：工单状态 订单状态, processing 处理中 ， reject 已拒绝 ， finish 已完成， exception 异常
RejectReason	String	拒绝原因 示例值：不符合规范
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId ）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 请求网络设备的通用服务订单详情

输入示例

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

{
"OrderId": "ord-25030712531166778"
}
```

输出示例

```
{
"Response": {
"BaseInfo": {
"ContactName": "zhangsan",
"ContactPhone": "13134567876",
"IdcName": "天津数据备份中心东区4栋DC",
"Instructions": "remark",
"PreAuthorization": true,
"ServiceLevel": 1
},
"DeviceSet": [
{
"DeviceType": "netDevice",
"IdcName": "天津数据备份中心东区4栋DC",
"IdcUnitName": "天津数据备份中心东区4栋DCM208",
"RackName": "M208-C08",
"Sn": "lhshenNet101"
}
],
"OrderStatus": "finish",
"RequestId": "8df49659-bc07-4364-ab1b-2c5981d3ab61",
"StepSet": [
{
"FinishTime": "2025-03-07 12:53:11",
"OwnerName": "zhangsan",
"StepName": "发起申请",
"StepStatus": "finish"
},
{
"FinishTime": "2025-03-07 13:02:37",
"OwnerName": "",
"StepName": "现场实施",
"StepStatus": "finish"
},
{
"FinishTime": "",
"OwnerName": "",
```

```
"StepName": "客户验证确认",
"StepStatus": "processing"
}
]
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

查询设备型号评估工单

最近更新时间：2025-04-25 01:16:39

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

查询设备型号评估工单详情

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeModelEvaluationWorkOrderDetail。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
OrderId	是	String	工单ID 示例值：ord-24111418091245911

3. 输出参数

参数名称	类型	描述
StepSet	Array of OrderStep	工单进度
BaseInfo	ModelEvaluationBaseInfo	工单详情
NetDeviceModelSet	Array of ModelVersionDetail	评估中的网络设备型号详情
ServerModelSet	Array of ModelVersionDetail	评估中的服务器型号详情
OrderStatus	String	订单状态, process 处理中 ， reject 已拒绝 ， finish 已完成，exception 异常 示例值：process
RejectReason	String	工单拒绝原因 示例值：不符合机房要求
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 查看服务器设备型号评估工单详情

输入示例

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

{
"OrderId": "ord-25030519265285037"
}
```

输出示例

```
{
"Response": {
"BaseInfo": {
"CampusName": "天津-武清",
"CustomerName": "某某公司",
"DeviceType": "server",
"Remark": "zhangsan"
},
"NetDeviceModelSet": [],
"OrderStatus": "finish",
"RequestId": "8ca708b5-695c-4c2d-be1a-39dd5c559710",
"ServerModelSet": [
{
"CheckResult": 1,
"ModelVersion": "LHSHEN T2121 S5-V1",
"OptionSet": [
{
"CompareType": "--",
"InputHint": "",
"InputInfo": "请注意节点数代表服务器子机数量，正常情况下标准服务器均为1",
"InputType": "dropDown",
"OptionKey": "devNode",
"OptionName": "节点数",
"OptionValue": "1",
"Standard": "--",
"StandardInfo": "--",
"ValueType": "number"
},
{
"CompareType": "--",
"InputHint": "填写设备高度（正整数，不带U）",
"InputInfo": "",
"InputType": "int",
"OptionKey": "devHeight",
```

```
"OptionName": "高度(u)",
"OptionValue": "22",
"Standard": "1u=44.45mm",
"StandardInfo": "1u=44.45mm",
"ValueType": "number"
},
{
"CompareType": "<",
"InputHint": "填写正整数",
"InputInfo": "",
"InputType": "int",
"OptionKey": "powerEnergy",
"OptionName": "功耗(W)",
"OptionValue": "330",
"Standard": "4400",
"StandardInfo": "单柜<4400/6600",
"ValueType": "number"
}
]
}
],
"StepSet": [
{
"FinishTime": "2025-03-05 19:26:52",
"OwnerName": "zhangsan",
"StepName": "发起申请",
"StepStatus": "finish"
},
{
"FinishTime": "2025-03-05 19:36:24",
"OwnerName": "lisi",
"StepName": "现场实施",
"StepStatus": "finish"
},
{
"FinishTime": "2025-03-05 19:36:24",
"OwnerName": "",
"StepName": "结单",
"StepStatus": "finish"
}
]
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

获取型号的填写模板

最近更新时间：2025-04-25 01:16:39

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

获取型号的填写模板

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：DescribeModelTemplate。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
DeviceType	是	String	型号类型，只支持传入 server 和 netDevice 示例值：server

3. 输出参数

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

4. 示例

示例1 获取型号的填写模板

输入示例

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

```
{
"DeviceType": "server"
}
```

输出示例

```
{
"Response": {
"RequestId": "1cb9293d-5bd1-4e2d-9dd3-d289b0d6c113",
"TemplateDetail": [
{
"CompareType": "--",
"InputHint": "",
"InputInfo": "请注意节点数代表服务器子机数量，正常情况下标准服务器均为1",
"InputType": "dropDown",
"OptionKey": "DevNode",
"OptionName": "节点数",
"OptionValueSet": [
{
"OptionValue": "1",
"Selected": true
},
{
"OptionValue": "4",
"Selected": false
}
],
"Standard": "--",
"StandardInfo": "--",
"ValueType": "number"
},
{
"CompareType": "--",
"InputHint": "填写设备高度（正整数，不带U）",
"InputInfo": "",
"InputType": "int",
"OptionKey": "DevHeight",
"OptionName": "高度(u)",
"OptionValueSet": [],
"Standard": "1u=44.45mm",
"StandardInfo": "1u=44.45mm",
"ValueType": "number"
},
{
"CompareType": "<",
"InputHint": "填写正整数",
"InputInfo": "",
"InputType": "int",
"OptionKey": "PowerEnergy",
```

```
"OptionName": "功耗(W)",
"OptionValueSet": [],
"Standard": "4400",
"StandardInfo": "单柜<4400/6600",
"ValueType": "number"
}
]
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

获取型号和版本列表

最近更新时间：2025-04-25 01:16:38

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

获取用户的型号和对应的版本数量

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：DescribeModelVersionList。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
DeviceType	是	String	型号类型，只支持传入 netDevice 和 server 示例值：netDevice
Filters.N	否	Array of Filter	model-name 型号名称 类型：String 必选：否 示例值：[{"Values":['TEST',],"Name":"model-name"}]
Checked	否	Boolean	是否已评估 示例值：true
CampusId	否	Integer	园区ID，当 Checked 参数传 True 时，该参数必须传值 示例值：1
ModelName	否	String	型号关键字，可以实现模糊匹配搜索功能 示例值：dell

3. 输出参数

参数名称	类型	描述
ModelVersionSet	Array of ModelVersionCount	型号和对应的版本数量 示例值：[{"DevModel":"ASA 5555","VersionCount":1}]
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 获取型号和对应版本数量

输入示例

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

{
"DeviceType": "server"
}
```

输出示例

```
{
"Response": {
"ModelVersionSet": [
{
"DevModel": "Test",
"VersionCount": 1
}
],
"RequestId": "d8d199be-bde9-4f2b-bcf7-3a979233b66a"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

查询人员到访工单详情

最近更新时间：2025-04-25 01:16:38

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

查询人员到访工单详情

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribePersonnelVisitWorkOrderDetail。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
OrderId	是	String	工单ID 示例值：ord-24112819463984820

3. 输出参数

参数名称	类型	描述
StepSet	Array of OrderStep	工单进度
BaseInfo	PersonnelVisitBaseInfo	工单详情
PersonnelSet	Array of Personnel	到访人员详情
OrderStatus	String	工单状态 订单状态, processing 处理中 ， reject 已拒绝 ， finish 已完成，exception 异常 示例值：processing
RejectReason	String	拒绝原因 示例值：不符合要求
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 获取人员到访

输入示例

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

{
"OrderId": "ord-25030712514742548"
}
```

输出示例


```
{
"Response": {
"BaseInfo": {
"EnterEndTime": "2025-03-21 00:00:00",
"EnterStartTime": "2025-03-20 00:00:00",
"IdcName": "天津数据备份中心东区3栋DC",
"IdcUnitNameList": [
"天津数据备份中心东区3栋DCM202"
],
"VisitReason": [
"OTHER"
],
"VisitRemark": "参观"
},
"OrderStatus": "processing",
"PersonnelSet": [
{
"Company": "1",
"Email": "",
"IDCardNumber": "100000000000000005",
"IDCardType": "IDENTITY_CARD",
"LanguageType": "CHINESE",
"Name": "1",
"Position": "",
"TelNumber": "13134567876",
"Wechat": ""
}
],
"RequestId": "50904c59-ffef-4fef-b504-2c637f98906a",
"StepSet": [
{
"FinishTime": "2025-03-07 12:51:47",
"OwnerName": "zhangsan",
"StepName": "发起申请",
"StepStatus": "finish"
},
{
```

```
"FinishTime": "",
"OwnerName": "zhangsan",
"StepName": "数经助理审批",
"StepStatus": "processing"
}
]
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

查询机位状态的汇总数据

最近更新时间：2025-04-25 01:16:38

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

获取机架总数及各状态对应的数量汇总

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：DescribePositionStatusSummary。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
Filters.N	否	Array of Filter	• rack-id 按照【机架ID】进行过滤。例如：15082。 类型：String 必选：否 • rack-name 按照【机架名称】进行过滤，机架名称例如：M301-E10。 类型：String 必选：否 • idc-id 按照【机房ID】进行过滤，机房ID例如：159。 类型：String 必选：否 • idc-unit-id 按照【机房管理单元ID】进行过滤，机房管理ID例如：568。 类型：String 必选：否 • position-status 按照【机位状态】进行过滤，机位状态只包含以下四种：机位状态,0 空闲,1 已用,2 不可用,3 预占用,4 预留，例如： 0。 类型：String 必选：否 • op-status

参数名称	必选	类型	描述
			按照【操作状态】进行过滤，操作状态只包含两种：FINISH 操作完成，PENDING 操作中，例如： PENDING。 类型：String 必选：否

3. 输出参数

参数名称	类型	描述
Total	Integer	总数 示例值：10
StatusCountSet	Array of PositionStatusItem	状态及对应数量
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 获取机位数据汇总

输入示例

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

{}
```

输出示例

```
{
"Response": {
"RequestId": "54c5a2de-513a-4d98-93c8-c0a682364495",
"StatusCountSet": [
{
"Count": 7442,
"PositionStatus": 0
},
{
"Count": 29447,
"PositionStatus": 1
},
{
"Count": 313,
"PositionStatus": 2
```

```
},
{
"Count": 3,
"PositionStatus": 3
},
{
"Count": 0,
"PositionStatus": 4
}
],
"Total": 37205
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

查询工单列表

最近更新时间：2025-04-25 01:16:38

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

查询工单列表

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：DescribeWorkOrderList。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
Filters.N	否	Array of Filter	过滤条件。支持：service-type、order-type、order-status、order-id
SnList.N	否	Array of String	通过sn过滤工单，上限10个 示例值：[td123456]
Offset	否	Integer	起始 示例值：0
Limit	否	Integer	长度 示例值：20

3. 输出参数

参数名称	类型	描述
TotalCount	Integer	总数 示例值：2
WorkOrderSet	Array of WorkOrderData	查询结果
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 查询工单列表

输入示例

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

{
"Limit": 1
}
```

输出示例

```
{
"Response": {
"RequestId": "646158b8-b20d-4843-a31a-883b78246cd8",
"TotalCount": 3247,
"WorkOrderSet": [
{
"CreateTime": "2025-03-07 11:11:59",
"Creator": "godwin2@test.com",
"OrderStatus": "processing",
"OrderType": "itCommon",
"ServiceType": "serverCommon",
"WorkOrderId": "ord-25030711115960582"
}
]
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

查询工单统计信息

最近更新时间：2025-04-25 01:16:38

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

工单统计数据查询

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：DescribeWorkOrderStatistics。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。

3. 输出参数

参数名称	类型	描述
TotalNum	Integer	总工单数量 示例值：100
ProcessingNum	Integer	进行中的工单数量 示例值：20
ConfirmingNum	Integer	待确认的工单数量 示例值：11
FinishNum	Integer	完成的工单数量 示例值：10
RejectNum	Integer	拒绝的工单数量 示例值：5
ExceptionNum	Integer	异常的工单数量 示例值：6
CancelNum	Integer	客户取消的工单数量 示例值：2
CheckingNum	Integer	围笼进出审核的工单数量 示例值：3

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

4. 示例

示例1 查询工单状态对应的数量

输入示例

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

{}
```

输出示例

```
{
"Response": {
"CancelNum": 0,
"CheckingNum": 0,
"ConfirmingNum": 0,
"ExceptionNum": 2748,
"FinishNum": 297,
"ProcessingNum": 112,
"RejectNum": 91,
"RequestId": "4c2a84c1-61ab-4f4c-b3b3-b38d7e6af4fc",
"TotalNum": 3248
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

获取机房内可用的型号列表

最近更新时间：2025-04-25 01:16:40

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

获取机房内可用的型号列表

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：DescribeAvailableModelList。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
IdcId	是	Integer	机房ID 示例值：159
DeviceType	是	String	server 服务器，netDevice 网络设备 示例值：设备类型

3. 输出参数

参数名称	类型	描述
ModelVersionSet	Array of AvailableModelVersion	机房内可用的设备型号及版本列表 示例值：[{"ModelVersion": "IBM X3650 M5-V1", "DevHeight": "2", "DeviceType": "server"}]
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 查看客户可用的服务器型号列表

输入示例

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

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

{
"IdcId": 373,
"DeviceType": "server"
}
```

输出示例


```
{
"Response": {
"ModelVersionSet": [
{
"DevHeight": "2",
"DeviceType": "server",
"ModelVersion": "SFT1502-III-V1"
},
{
"DevHeight": "30",
"DeviceType": "server",
"ModelVersion": "lhshen server test 03-V1"
},
{
"DevHeight": "2",
"DeviceType": "server",
"ModelVersion": "DELL R740-T1-V1"
},
{
"DevHeight": "2",
"DeviceType": "server",
"ModelVersion": "Dell R740-T001-V1"
},
{
"DevHeight": "4",
"DeviceType": "server",
"ModelVersion": "LD-002-V1"
},
{
"DevHeight": "2",
"DeviceType": "server",
"ModelVersion": "LD-SERVER-001-V1"
},
{
"DevHeight": "23",
"DeviceType": "server",
"ModelVersion": "BM X3650 M5003-V1"
},
```

```
{
"DevHeight": "22",
"DeviceType": "server",
"ModelVersion": "IBM X3650 M5007-V1"
},
{
"DevHeight": "69",
"DeviceType": "server",
"ModelVersion": "lihwli0217001-V2"
},
{
"DevHeight": "23",
"DeviceType": "server",
"ModelVersion": "234-V1"
},
{
"DevHeight": "1",
"DeviceType": "server",
"ModelVersion": "QW LD003 V1-V1"
},
{
"DevHeight": "1",
"DeviceType": "server",
"ModelVersion": "QW LD003 V2-V1"
},
{
"DevHeight": "23",
"DeviceType": "server",
"ModelVersion": "lihwli02250001-V1"
},
{
"DevHeight": "11",
"DeviceType": "server",
"ModelVersion": "lhshen-test-V1"
}
],
"RequestId": "5bc61ca8-aa3c-4307-a239-873fbb8e5b18"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

创建服务器型号

最近更新时间：2025-04-25 01:16:40

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

新增服务器设备型号

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：CreateServerModel。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
ModelDetail	是	ServerModel	设备型号详情

3. 输出参数

参数名称	类型	描述
DevModel	String	型号名称 示例值：SFT1502-III
Version	String	版本 示例值：V1
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 创建新服务器型号

输入示例

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

```
{
"ModelDetail": {
"DevModel": "XXX-TEST",
"DevNode": "1",
"DevHeight": "9",
"PowerEnergy": "500",
"PowerportType": "C14",
"PowermoduleNum": "2",
"InwindPosition": "前",
"OutwindPosition": "后",
"NetportPosition": "设备后端",
"DevWidth": "<370",
"DevDepth": "≤780",
"DevWeight": "≤50",
"PowerModule": "兼容直流270DC和交流",
"PowermodulePosition": "设备后端",
"NetportType": "电口",
"NetSpeed": "千兆"
}
}
```


输出示例


```
{
"Response": {
"DevModel": "XXX-TEST",
"RequestId": "21f5aef1-4251-4f47-81de-aaf03c4cbfa5",
"Version": "V1"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

创建临时设备退出工单

最近更新时间：2025-04-25 01:16:40

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

创建临时设备退出工单

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：CreateSpeciallyQuitWorkOrder。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
IdcId	是	Integer	机房id 示例值：325
DeviceType	是	String	设备类型：otherDevice。此接口只支持其他设备 示例值：otherDevice
HandoverMethod	是	String	交接方式 1.物流上门收货 2.客户上门自提 示例值：1
LogisticsReceipt	否	LogisticsReceipt	物流上门收货必传
CustomerReceipt	否	CustomerReceipt	客户上门自提必传
Remark	否	String	备注信息 示例值：备注信息
OtherDeviceList.N	否	Array of OtherDevReceivingInfo	当设备类型为otherDevice，此参数必传

3. 输出参数

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

4. 示例

示例1 创建临时设备退出工单

输入示例

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

{
"IdcId": 159,
"DeviceType": "otherDevice",
"HandoverMethod": "2",
"CustomerReceipt": {
"PickUpStuff": "张三",
"PickUpStuffContact": "16611111111",
"PickUpStuffIDCard": "632525198112198139",
"PickUpTime": "2025-12-11 12:00:00"
},
"Remark": "测试xzh111",
"OtherDeviceList": [
{
"DeviceSn": "20250403chc1222",
"TypeName": "移动硬盘",
"Manufacturer": "华为",
"HardwareMemo": "xx设备第二个硬盘"
},
{
"DeviceSn": "20240101chc0008",
"TypeName": "其他",
"HardwareMemo": "xxx设备第3个xx"
}
]
}
```

输出示例

```
{
"Response": {
"RequestId": "f8fedcd3-198f-4ac5-be69-1b54ae62601d",
"WorkOrderSet": [
{
"OrderType": "quit",
"ServiceType": "quit",
"WorkOrderId": "ord-25041015202483289"
},
{
```

```
"OrderType": "personnelVisit",
"ServiceType": "quit",
"WorkOrderId": "ord-25041015202481186"
}
]
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

确认通用服务工单

最近更新时间：2025-04-25 01:16:41

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

确认通用服务工单

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：ConfirmCommonServiceWorkOrder。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
OrderId	是	String	工单ID 示例值：ord-24120219411076552

3. 输出参数

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

4. 示例

示例1 确认通用服务

输入示例

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

{
```

```
"OrderId": "ord-24122617232912748"
}
```

输出示例

```
{
"Response": {
"RequestId": "56d72661-0f9d-40ab-b3ca-a4adb352996e"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

创建通用服务工单

最近更新时间：2025-04-25 01:16:41

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

创建通用工单

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：CreateCommonServiceWorkOrder。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
DevicePositionSet.N	是	Array of DevicePosition	设备及位置信息列表
ServiceLevel	是	Integer	服务级别，只支持传入 1、2、3，分别对应 L1、L2、L3 示例值：1
PreAuthorization	是	Boolean	操作预授权 示例值：true
ContactName	是	String	业务联系人 示例值：张三
ContactPhone	是	String	联系人电话 示例值：13800138000
DeviceType	是	String	设备类型 server 服务器，netDevice 网络设备 示例值：server
Instructions	是	String	操作说明 示例值：检查五台设备

3. 输出参数

参数名称	类型	描述
WorkOrderSet	Array of WorkOrderTinyInfo	创建的工单信息 示例值：[{ "WorkOrderId": "ord-24122317200212924", "ServiceType": "itCommon", "OrderType": "itCommon" }]

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

4. 示例

示例1 提交服务器类型的通用服务工单

输入示例

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

{
"DevicePositionSet": [
{
"Sn": "217404176",
"RackName": "M203-B14",
"IdcUnitId": 2555,
"PositionCode": 1
}
],
"ServiceLevel": 1,
"PreAuthorization": true,
"ContactName": "张三",
"ContactPhone": "13800138000",
"DeviceType": "server",
"Instructions": "检查五台设备"
}
```

输出示例

```
{
"Response": {
"RequestId": "7600c206-c7c2-40f8-bea8-195a2efa8c7a",
"WorkOrderSet": [
{
"OrderType": "itCommon",
"ServiceType": "serverCommon",
"WorkOrderId": "ord-25030711115960582"
}
]
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

创建设备型号评估工单

最近更新时间：2025-04-25 01:16:41

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

创建设备型号评估工单

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：CreateModelEvaluationWorkOrder。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
ModelSet.N	是	Array of ModelVersion	设备名称及型号 示例值：[{ "DevModel": "SFT1502-III", "Version": "V1" }]
CampusId	是	Integer	园区ID 示例值：6
DeviceType	是	String	只支持传入 server 和 netDevice 示例值：server
Remark	否	String	备注 示例值：备注

3. 输出参数

参数名称	类型	描述
WorkOrderSet	Array of WorkOrderTinyInfo	创建的工单信息 示例值：[{ "WorkOrderId": "ord-24122317200212924", "ServiceType": "modelEvaluation", "OrderType": "modelEvaluation" }]
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 创建服务器类型的设备型号评估工单

输入示例

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

{
"ModelSet": [
{
"DevModel": "IBM X3650 M5 013",
"Version": "V1"
}
],
"CampusId": 163,
"DeviceType": "server",
"Remark": "服务器型号评估"
}
```

输出示例

```
{
"Response": {
"RequestId": "9bee9bdf-b631-414d-8d9b-20bdaddb87a2",
"WorkOrderSet": [
{
"OrderType": "modelEvaluation",
"ServiceType": "modelEvaluation",
"WorkOrderId": "ord-25030711220788330"
}
]
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

创建设备搬迁工单

最近更新时间：2025-04-25 01:16:41

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

创建设备搬迁工单

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：CreateMovingWorkOrder。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
IdcId	是	Integer	机房id 示例值：325
DeviceType	是	String	设备类型，server, netDevice 示例值：server
WithPowerOn	是	Boolean	上架后是否开电 示例值：false
DeviceMovingList.N	是	Array of DeviceRackOn	设备搬迁信息列表
Remark	否	String	备注 示例值：备注：搬迁

3. 输出参数

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

4. 示例

示例1 创建设备搬迁工单

输入示例

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

{
"IdcId": 373,
"DeviceType": "server",
"WithPowerOn": false,
"DeviceMovingList": [
{
"DeviceSn": "LD-SERVER-20250124-002",
"DstRackName": "M302-J01",
"DstPositionCode": "20"
}
]
}
```

输出示例

```
{
"Response": {
"RequestId": "bd5139d4-b3fa-41ef-bf7a-9e8166975903",
"WorkOrderSet": [
{
"OrderType": "rackOff",
"ServiceType": "moving",
"WorkOrderId": "ord-25030714334568923"
},
{
"OrderType": "rackOn",
"ServiceType": "moving",
"WorkOrderId": "ord-25030714334565831"
}
]
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

创建人员到访工单

最近更新时间：2025-04-25 01:16:41

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

创建人员到访工单

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：CreatePersonnelVisitWorkOrder。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
PersonnelSet.N	是	Array of Personnel	到访人员信息
IdcId	是	Integer	机房 ID 示例值：169
IdcUnitIdSet.N	是	Array of Integer	机房管理单元列表 示例值：[15802, 15803]
EnterStartTime	是	String	到访开始时间 示例值：2024-10-27 00:25:56
EnterEndTime	是	String	到访结束时间 示例值：2024-10-27 00:25:56
VisitReason.N	是	Array of String	到访原因，映射关系：IT_OPERATION IT运维 OTHER 其他 示例值：["OTHER"]
VisitRemark	是	String	到访说明 示例值：设备测试

3. 输出参数

参数名称	类型	描述
WorkOrderSet	Array of WorkOrderTinyInfo	创建的工单信息 示例值：[{ "WorkOrderId": "ord-24122317200212924", "ServiceType": "personnelVisit", "OrderType": "personnelVisit" }]

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

4. 示例

示例1 创建人员到访工单

输入示例

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

{
"PersonnelSet": [
{
"IDCardNumber": "****",
"IDCardType": "IDENTITY_CARD",
"Company": "某某公司",
"LanguageType": "CHINESE",
"Name": "张三",
"TelNumber": "13800138000",
"Position": "工程师",
"Wechat": "zhangsan_wechat",
"Email": "zhangsan@example.com"
}
],
"IdcId": 373,
"IdcUnitIdSet": [
2555
],
"EnterStartTime": "2025-03-08 17:25:04",
"EnterEndTime": "2025-03-10 17:25:04",
"VisitReason": [
"IT_OPERATION"
],
"VisitRemark": "需要参观"
}
```

输出示例

```
{
"Response": {
"RequestId": "289288d7-d662-4b66-b4ef-d18d8913ca18",
"WorkOrderSet": [
{
```

```
"OrderType": "personnelVisit",
"ServiceType": "personnelVisit",
"WorkOrderId": "ord-25030711303576000"
}
]
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

创建设备关电工单

最近更新时间：2025-04-25 01:16:40

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

创建设备关电工单

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：CreatePowerOffWorkOrder。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
IdcId	是	Integer	机房id 示例值：325
DeviceType	是	String	设备类型，server, netDevice 示例值：server
IsPowerOffConfirm	是	String	关电确认，1.授权时关电 2.关电前需要确认 示例值：1
DeviceSnList.N	是	Array of String	设备sn列表 示例值：['chc001','chc002','chc003']
PowerOffConfirmInfo	否	PowerOffConfirm	关电前需要确认时必填
Remark	否	String	备注 示例值：备注：关电

3. 输出参数

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

4. 示例

示例1 创建设备关电工单

输入示例

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

{
"IdcId": 2005,
"DeviceType": "server",
"IsPowerOffConfirm": "1",
"DeviceSnList": [
"lihwli02130003"
]
}
```

输出示例

```
{
"Response": {
"RequestId": "f60e609f-eaa9-43c2-ba21-fcbc9e321efe",
"WorkOrderSet": [
{
"OrderType": "powerOff",
"ServiceType": "powerOff",
"WorkOrderId": "ord-25030714181161269"
}
]
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

创建设备开电工单

最近更新时间：2025-04-25 01:16:40

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

创建设备开电工单

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：CreatePowerOnWorkOrder。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
IdcId	是	Integer	机房id 示例值：325
DeviceType	是	String	设备类型，server, netDevice 示例值：server
DeviceSnList.N	是	Array of String	设备sn列表 示例值：['chc001','chc002']

3. 输出参数

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

4. 示例

示例1 创建设备开电工单

输入示例

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

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

{
"IdcId": 373,
"DeviceType": "server",
"DeviceSnList": [
"TEN948P04K"
]
}
```

输出示例

```
{
"Response": {
"RequestId": "18a2463c-d31c-4125-8d28-7c5f018e7440",
"WorkOrderSet": [
{
"OrderType": "powerOn",
"ServiceType": "powerOn",
"WorkOrderId": "ord-25030711320087815"
}
]
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

创建设备退出工单

最近更新时间：2025-04-25 01:16:40

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

创建设备退出工单

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：CreateQuitWorkOrder。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
IdcId	是	Integer	机房id 示例值：325
DeviceType	是	String	设备类型，server, netDevice, otherDevice 示例值：server
StuffOption	是	String	下架选择 1.自行解决 2.由腾讯IDC负责 示例值：1
IsPowerOffConfirm	是	String	关电确认 1.授权时关电 2.关电前需要确认 示例值：1
DeviceSnList.N	是	Array of String	设备sn列表 示例值：['chc001','chc002']
HandoverMethod	是	String	交接方式 1.物流上门收货 2.客户上门自提 示例值：1
SelfOperationInfo	否	SelfOperation	自行解决必填
PowerOffConfirmInfo	否	PowerOffConfirm	关电前需要确认时必填
Remark	否	String	备注 示例值：备注信息
LogisticsReceipt	否	LogisticsReceipt	物流上门收货必传
CustomerReceipt	否	CustomerReceipt	客户上门自提必传

3. 输出参数

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

4. 示例

示例1 创建设备退出工单

输入示例

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

{
"IdcId": 2005,
"DeviceType": "server",
"StuffOption": "2",
"IsPowerOffConfirm": "1",
"DeviceSnList": [
"lihwli021718420010"
],
"HandoverMethod": "2",
"CustomerReceipt": {
"PickUpStuff": "张三",
"PickUpStuffContact": "15311111111",
"PickUpStuffIDCard": "100000000000000005",
"PickUpTime": "2025-03-08 12:00:00"
}
}
```

输出示例

```
{
"Response": {
"RequestId": "ece01139-3175-4f9e-8c15-3a513839c405",
"WorkOrderSet": [
{
"OrderType": "powerOff",
"ServiceType": "quit",
"WorkOrderId": "ord-25030714282926976"
},
{
"OrderType": "rackOff",
```

```
"ServiceType": "quit",
"WorkOrderId": "ord-25030714282967608"
},
{
"OrderType": "quit",
"ServiceType": "quit",
"WorkOrderId": "ord-25030714282957052"
},
{
"OrderType": "personnelVisit",
"ServiceType": "quit",
"WorkOrderId": "ord-25030714282880371"
}
]
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

创建设备下架工单

最近更新时间：2025-04-25 01:16:40

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

创建设备下架工单

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：CreateRackOffWorkOrder。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
IdcId	是	Integer	机房id 示例值：325
DeviceType	是	String	设备类型，server, netDevice 示例值：server
StuffOption	是	String	下架人员 1.自行解决 2.由腾讯IDC负责 示例值：1
IsPowerOffConfirm	是	String	关电确认 1.授权时关电 2.关电前需要确认 示例值：1
DeviceSnList.N	是	Array of String	设备sn列表 示例值：['chc001','chc002']
SelfOperationInfo	否	SelfOperation	自行解决必填
PowerOffConfirmInfo	否	PowerOffConfirm	关电前需要确认时必填
Remark	否	String	备注 示例值：备注信息

3. 输出参数

参数名称	类型	描述
WorkOrderSet	Array of WorkOrderTinyInfo	创建的工单信息

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

4. 示例

示例1 创建设备下架工单

输入示例

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

{
"IdcId": 2005,
"DeviceType": "server",
"StuffOption": "2",
"IsPowerOffConfirm": "1",
"DeviceSnList": [
"lihwli021718420012"
]
}
```

输出示例

```
{
"Response": {
"RequestId": "5b09296a-8e17-4033-846e-79acaf3bc809",
"WorkOrderSet": [
{
"OrderType": "powerOff",
"ServiceType": "rackOff",
"WorkOrderId": "ord-25030714251078382"
},
{
"OrderType": "rackOff",
"ServiceType": "rackOff",
"WorkOrderId": "ord-25030714251022115"
}
]
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

创建设备上架工单

最近更新时间：2025-04-25 01:16:40

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

创建设备上架工单

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：CreateRackOnWorkOrder。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
IdcId	是	Integer	机房id 示例值：325
DeviceType	是	String	设备类型，server, netDevice 示例值：server
StuffOption	是	String	上架人员 1.自行解决 2.由腾讯IDC负责 示例值：1
WithPowerOn	是	Boolean	上架后是否开电 示例值：false
DeviceRackOnList.N	是	Array of DeviceRackOn	服务器收货列表
SelfOperationInfo	否	SelfOperation	自行解决必填

3. 输出参数

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

4. 示例

示例1 创建设备上架工单

输入示例

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

{
"IdcId": 373,
"DeviceType": "server",
"StuffOption": "2",
"WithPowerOn": false,
"DeviceRackOnList": [
{
"DeviceSn": "lihwli10225021",
"DstRackName": "M302-J01",
"DstPositionCode": "7"
}
]
}
```

输出示例

```
{
"Response": {
"RequestId": "4fd6eff1-2613-48dd-be51-a74ac082ff34",
"WorkOrderSet": [
{
"OrderType": "rackOn",
"ServiceType": "rackOn",
"WorkOrderId": "ord-25030711311423914"
}
]
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

创建设备收货工单

最近更新时间：2025-05-08 01:17:32

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

创建设备收货工单

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：CreateReceivingWorkOrder。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
IdcId	是	Integer	机房id 示例值：325
DeviceType	是	String	设备类型，server, netDevice, wire, otherDevice 示例值：server
EntryTime	是	String	进入时间 示例值：2024-11-30 00:00:00
ReceivingOperation	是	String	1.收货-仅核对外包装完整和数量，不开箱 2.验收-开箱核对型号SN一致 示例值：1
IsExpressDelivery	否	Boolean	是否快递寄件 示例值：false
ExpressInfo	否	ExpressDelivery	快递寄件信息,快递寄件必填
Remark	否	String	备注 示例值：备注信息
ServerDeviceList.N	否	Array of ServerReceivingInfo	服务器收货列表。最大值：200
NetDeviceList.N	否	Array of NetReceivingInfo	网络设备收货列表
WireDeviceList.N	否	Array of WireReceivingInfo	线材收货列表
OtherDeviceList.N	否	Array of OtherDevReceivingInfo	其他设备收货列表

参数名称	必选	类型	描述
WithRackOn	否	Boolean	收货后自动上架。此参数为true时，后台会自动提设备上架单 示例值：true
DeviceRackOnList.N	否	Array of DeviceRackOn	设备上架信息。当WithRackOn为true此参数必传，且sn需要和收货的列表一致
StuffOption	否	String	上架人员 1.自行解决 2.由腾讯IDC负责 示例值：2
SelfOperationInfo	否	SelfOperation	自行解决信息。当StuffOption为1时，此参数必填
WithPowerOn	否	Boolean	上架后自动开电。此参数为true时，后台会自动提设备开电单 示例值：true

3. 输出参数

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

4. 示例

示例1 创建设备收货工单

输入示例

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

{
"IdcId": 159,
"DeviceType": "server",
"EntryTime": "2025-03-08 00:00:00",
"ReceivingOperation": "1",
"IsExpressDelivery": false,
"ServerDeviceList": [
{
"DeviceSn": "chc20250308xxx001",
"ModelVersion": "DELL R740-T1-V1"
}
]
}
```

输出示例

```
{
"Response": {
"RequestId": "e1006a5c-4a9f-463e-963a-7f6ba01edffe",
"WorkOrderSet": [
{
"OrderType": "receiving",
"ServiceType": "receiving",
"WorkOrderId": "ord-25030711283032499"
}
]
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

查询用户可用的工单列表

最近更新时间：2025-04-25 01:16:38

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

获取用户可用的工单类型

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：DescribeWorkOrderTypes。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。

3. 输出参数

参数名称	类型	描述
CollectedWorkOderTypeSet	Array of WorkOrderTypeDetail	已收藏的工单类型列表
WorkOrderFamilySet	Array of WorkOrderFamilyDetail	全部工单类型列表
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 请求示例

输入示例

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

```
{}
```

输出示例

```
{
"Response": {
"CollectedWorkOderTypeSet": [
{
"CollectFlag": true,
"SlaMessage": "2个工作日",
"WorkOrderDescription": "客户设备托管到腾讯IDC机房，为了确保设备符合机房的上架标准， 在入场前，需要提前进行托管设备入场的评估。",
"WorkOrderFamily": "设备服务",
"WorkOrderName": "托管设备型号评估",
"WorkOrderType": "modelEvaluation"
},
{
"CollectFlag": true,
"SlaMessage": "2 个工作日",
"WorkOrderDescription": "提供现场外包资产管理员，进行现场物资的收货和验收操作。\n",
"WorkOrderFamily": "设备服务",
"WorkOrderName": "设备收货",
"WorkOrderType": "receiving"
},
{
"CollectFlag": true,
"SlaMessage": "2 个工作日",
"WorkOrderDescription": "提供现场外包工程师，将指定设备上到机位上。 ",
"WorkOrderFamily": "设备服务",
"WorkOrderName": "设备上架",
"WorkOrderType": "rackOn"
},
{
"CollectFlag": true,
"SlaMessage": "最快 2 个工作日",
"WorkOrderDescription": "提供设备的开电操作。机架的开电由IDC平台部数经判断，自行发起。\n",
"WorkOrderFamily": "设备服务",
"WorkOrderName": "设备开电",
"WorkOrderType": "powerOn"
},
{
"CollectFlag": true,
"SlaMessage": "最快 1 个工作日",
"WorkOrderDescription": "根据客户要求， 在同IDC内对指定设备进行搬迁操作。",
"WorkOrderFamily": "设备服务",
"WorkOrderName": "机房内搬迁",
"WorkOrderType": "moving"
}
```

```
],
"RequestId": "3dad52cf-67c5-4eed-b191-e4f542ce4d80",
"WorkOrderFamilySet": [
{
"WorkOrderFamily": "设备服务",
"WorkOrderTypeSet": [
{
"CollectFlag": true,
"SlaMessage": "2个工作日",
"WorkOrderDescription": "客户设备托管到腾讯IDC机房，为了确保设备符合机房的上架标准， 在入场前，需要提前进行托管设备入场的评估。",
"WorkOrderFamily": "设备服务",
"WorkOrderName": "托管设备型号评估",
"WorkOrderType": "modelEvaluation"
},
{
"CollectFlag": true,
"SlaMessage": "2 个工作日",
"WorkOrderDescription": "提供现场外包资产管理员，进行现场物资的收货和验收操作。\n",
"WorkOrderFamily": "设备服务",
"WorkOrderName": "设备收货",
"WorkOrderType": "receiving"
},
{
"CollectFlag": true,
"SlaMessage": "2 个工作日",
"WorkOrderDescription": "提供现场外包工程师，将指定设备上到机位上。 ",
"WorkOrderFamily": "设备服务",
"WorkOrderName": "设备上架",
"WorkOrderType": "rackOn"
},
{
"CollectFlag": true,
"SlaMessage": "最快 2 个工作日",
"WorkOrderDescription": "提供设备的开电操作。机架的开电由IDC平台部数经判断，自行发起。\n",
"WorkOrderFamily": "设备服务",
"WorkOrderName": "设备开电",
"WorkOrderType": "powerOn"
},
{
"CollectFlag": true,
"SlaMessage": "最快 1 个工作日",
"WorkOrderDescription": "根据客户要求， 在同IDC内对指定设备进行搬迁操作。",
"WorkOrderFamily": "设备服务",
"WorkOrderName": "机房内搬迁",
"WorkOrderType": "moving"
},
{
"CollectFlag": false,
"SlaMessage": "最快 2 个工作日",
```

```
"WorkOrderDescription": "提供设备的关电操作。\n",
"WorkOrderFamily": "设备服务",
"WorkOrderName": "设备关电",
"WorkOrderType": "powerOff"
},
{
"CollectFlag": false,
"SlaMessage": "最快当天完成",
"WorkOrderDescription": "提供现场外包工程师，将指定设备从机位上下架。",
"WorkOrderFamily": "设备服务",
"WorkOrderName": "设备下架",
"WorkOrderType": "rackOff"
},
{
"CollectFlag": false,
"SlaMessage": "1 个工作日",
"WorkOrderDescription": "客户将设备等资产迁出腾讯机房， 腾讯提供指定设备的交接服务， 不包含物流服务，不包含包装、搬车和装运服务。",
"WorkOrderFamily": "设备服务",
"WorkOrderName": "退出交接",
"WorkOrderType": "quit"
},
{
"CollectFlag": false,
"SlaMessage": "最快 2 个工作日",
"WorkOrderDescription": "提供机架的开电操作。",
"WorkOrderFamily": "设备服务",
"WorkOrderName": "机架开电",
"WorkOrderType": "rackPowerOn"
},
{
"CollectFlag": false,
"SlaMessage": "最快 2 个工作日",
"WorkOrderDescription": "提供机架的关电操作。",
"WorkOrderFamily": "设备服务",
"WorkOrderName": "机架关电",
"WorkOrderType": "rackPowerOff"
}
]
},
{
"WorkOrderFamily": "人员服务",
"WorkOrderTypeSet": [
{
"CollectFlag": false,
"SlaMessage": "最快 1 个工作日",
"WorkOrderDescription": "非机房常驻运维及授权人员，因运维需要进入机房，需提交人员到访申请。",
"WorkOrderFamily": "人员服务",
"WorkOrderName": "人员到访",
"WorkOrderType": "personnelVisit"
```

```
}
]
},
{
"WorkOrderFamily": "通用服务",
"WorkOrderTypeSet": [
{
"CollectFlag": false,
"SlaMessage": "最快 30min 响应",
"WorkOrderDescription": "提供网络设备的设备重启、状态查看等通用服务受理",
"WorkOrderFamily": "通用服务",
"WorkOrderName": "网络设备通用服务",
"WorkOrderType": "netDeviceCommon"
},
{
"CollectFlag": false,
"SlaMessage": "最快 30min 响应",
"WorkOrderDescription": "提供服务器的设备重启、插拔硬盘、重装系统、状态查看等通用服务受理",
"WorkOrderFamily": "通用服务",
"WorkOrderName": "服务器通用服务",
"WorkOrderType": "serverCommon"
}
]
}
]
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

修改工单类型的收藏标识

最近更新时间：2025-04-25 01:16:38

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

如果当前该工单类型是收藏状态，调用接口后变成未收藏状态，如果是未收藏状态，调用该接口变为收藏状态

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：ModifyWorkOrderTypeCollectFlag。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
WorkOrderType	是	String	工单类型的唯一英文标识 示例值：modelEvaluation

3. 输出参数

参数名称	类型	描述
CurrentCollectFlag	Boolean	工单类型当前的收藏状态 示例值：true
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 修改网络通用服务的收藏标识

输入示例

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

```
{
"WorkOrderType": "netDeviceCommon"
}
```

输出示例

```
{
"Response": {
"CurrentCollectFlag": true,
"RequestId": "14ec2fbf-8b7d-4b55-ba69-207d7ca6c15e"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

创建网络设备新型号

最近更新时间：2025-04-25 01:16:41

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

创建新的网络设备型号

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：CreateNetDeviceModel。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
ModelDetail	是	NetDeviceModel	网络设备型号

3. 输出参数

参数名称	类型	描述
DevModel	String	型号名称 示例值：Cisco IronPort 370
Version	String	版本号 示例值：V1
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 创建网络设备型号

输入示例

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

```
{
"ModelDetail": {
"DevModel": "XXX-test",
"DevWidth": "370－446",
"DevDepth": "≤1000",
"DevWeight": "≤50",
"MountEar": "否",
"AccordCCC": "是",
"PassNetwork": "是",
"PowerportType": "C13",
"PowerModule": "交流",
"PowermoduleNum": "2",
"PowermodulePosition": "设备后端",
"HighVoltageAdapt": "否",
"PowerEnergy": "350",
"InwindPosition": "前",
"OutwindPosition": "后",
"BusinessPortPosition": "后",
"LineManager": "否",
"DevHeight": "2"
}
}
```


输出示例


```
{
"Response": {
"DevModel": "XXX-test",
"RequestId": "24943157-d39d-44b7-ad4f-c9498f6a3b23",
"Version": "V1"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

资源相关接口
查询资源汇总

最近更新时间：2025-04-25 01:16:37

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

查询资源汇总

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：DescribeResourceUsage。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
Filters.N	否	Array of Filter	过滤条件 示例值：[{ "Values": [691], "Name": "idc-id" }, { "Values": [2725], "Name": "idc-unit-id" }]

3. 输出参数

参数名称	类型	描述
HostingServerCount	Integer	托管服务器数量 示例值：2
RentServerCount	Integer	租用服务器数量 示例值：2
NetDeviceCount	Integer	网络设备数量 示例值：2
RackTotalCount	Integer	机架总数 示例值：2
RackPowerOnCount	Integer	开电机架总数 示例值：2
PositionUsedCount	Integer	机位使用数量 示例值：2

参数名称	类型	描述
PositionTotalCount	Integer	机位总数 示例值：2
RackPowerOnRate	String	机架开电率，保留一位小数 示例值：69.2
PositionUsedRate	String	机位使用率， 示例值：69.2
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 获取资源汇总

输入示例

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

{}
```

输出示例

```
{
"Response": {
"HostingServerCount": 1367,
"NetDeviceCount": 893,
"PositionTotalCount": 37205,
"PositionUsedCount": 29447,
"PositionUsedRate": "79.1",
"RackPowerOnCount": 1796,
"RackPowerOnRate": "81.0",
"RackTotalCount": 2217,
"RentServerCount": 1146,
"RequestId": "5964d012-da09-4fb6-ac4d-8cf019f52a00"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

获取机位分布

最近更新时间：2025-04-25 01:16:37

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

获取机房管理单元的机位分布

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：DescribeRacksDistribution。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
IdcUnitId	是	Integer	机房管理单元ID 示例值：4261

3. 输出参数

参数名称	类型	描述
DistributionSet	Array of Distribution	机架的用量分布 示例值：[{ "RackNumber": "C", "RackUsageSet": [{ "RackId": 132316, "UsedNum": 1, "UnusedNum": 0, "RackShortName": "C05" }] }]
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 获取机架分布列表

输入示例

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

```
{
"IdcUnitId": 2555
}
```

输出示例


```
{
"Response": {
"DistributionSet": [
{
"RackNumber": "C",
"RackUsageSet": [
{
"RackId": 91831,
"RackShortName": "C01",
"TotalNum": 10,
"UnusedNum": 8,
"UsedNum": 2,
"UsedRate": 0.2
},
{
"RackId": 91840,
"RackShortName": "C04",
"TotalNum": 25,
"UnusedNum": 5,
"UsedNum": 20,
"UsedRate": 0.8
},
{
"RackId": 91843,
"RackShortName": "C05",
"TotalNum": 26,
"UnusedNum": 10,
"UsedNum": 16,
"UsedRate": 0.62
},
{
"RackId": 91846,
"RackShortName": "C06",
"TotalNum": 25,
"UnusedNum": 19,
"UsedNum": 6,
"UsedRate": 0.24
},
{
"RackId": 91849,
"RackShortName": "C07",
"TotalNum": 25,
```

```
"UnusedNum": 22,
"UsedNum": 3,
"UsedRate": 0.12
},
{
"RackId": 91852,
"RackShortName": "C08",
"TotalNum": 26,
"UnusedNum": 26,
"UsedNum": 0,
"UsedRate": 0
},
{
"RackId": 91855,
"RackShortName": "C09",
"TotalNum": 25,
"UnusedNum": 23,
"UsedNum": 2,
"UsedRate": 0.08
},
{
"RackId": 91858,
"RackShortName": "C10",
"TotalNum": 10,
"UnusedNum": 10,
"UsedNum": 0,
"UsedRate": 0
},
{
"RackId": 91867,
"RackShortName": "C13",
"TotalNum": 25,
"UnusedNum": 11,
"UsedNum": 14,
"UsedRate": 0.56
},
{
"RackId": 91870,
"RackShortName": "C14",
"TotalNum": 26,
"UnusedNum": 15,
"UsedNum": 11,
"UsedRate": 0.42
},
{
"RackId": 91873,
"RackShortName": "C15",
"TotalNum": 25,
"UnusedNum": 23,
"UsedNum": 2,
```

```
"UsedRate": 0.08
}
]
},
{
"RackNumber": "D",
"RackUsageSet": [
{
"RackId": 91885,
"RackShortName": "D01",
"TotalNum": 25,
"UnusedNum": 22,
"UsedNum": 3,
"UsedRate": 0.12
},
{
"RackId": 91894,
"RackShortName": "D04",
"TotalNum": 25,
"UnusedNum": 25,
"UsedNum": 0,
"UsedRate": 0
},
{
"RackId": 91897,
"RackShortName": "D05",
"TotalNum": 26,
"UnusedNum": 17,
"UsedNum": 9,
"UsedRate": 0.35
},
{
"RackId": 91900,
"RackShortName": "D06",
"TotalNum": 25,
"UnusedNum": 15,
"UsedNum": 10,
"UsedRate": 0.4
},
{
"RackId": 91903,
"RackShortName": "D07",
"TotalNum": 25,
"UnusedNum": 23,
"UsedNum": 2,
"UsedRate": 0.08
},
{
"RackId": 91906,
"RackShortName": "D08",
```

```
"TotalNum": 26,
"UnusedNum": 19,
"UsedNum": 7,
"UsedRate": 0.27
},
{
"RackId": 91909,
"RackShortName": "D09",
"TotalNum": 25,
"UnusedNum": 12,
"UsedNum": 13,
"UsedRate": 0.52
},
{
"RackId": 91912,
"RackShortName": "D10",
"TotalNum": 25,
"UnusedNum": 14,
"UsedNum": 11,
"UsedRate": 0.44
},
{
"RackId": 91915,
"RackShortName": "D11",
"TotalNum": 26,
"UnusedNum": 18,
"UsedNum": 8,
"UsedRate": 0.31
},
{
"RackId": 91918,
"RackShortName": "D12",
"TotalNum": 25,
"UnusedNum": 20,
"UsedNum": 5,
"UsedRate": 0.2
},
{
"RackId": 91921,
"RackShortName": "D13",
"TotalNum": 25,
"UnusedNum": 18,
"UsedNum": 7,
"UsedRate": 0.28
},
{
"RackId": 91924,
"RackShortName": "D14",
"TotalNum": 26,
"UnusedNum": 23,
```

```
"UsedNum": 3,
"UsedRate": 0.12
},
{
"RackId": 91927,
"RackShortName": "D15",
"TotalNum": 25,
"UnusedNum": 12,
"UsedNum": 13,
"UsedRate": 0.52
}
]
},
{
"RackNumber": "E",
"RackUsageSet": [
{
"RackId": 91948,
"RackShortName": "E04",
"TotalNum": 10,
"UnusedNum": 8,
"UsedNum": 2,
"UsedRate": 0.2
},
{
"RackId": 91951,
"RackShortName": "E05",
"TotalNum": 10,
"UnusedNum": 10,
"UsedNum": 0,
"UsedRate": 0
}
]
}
],
"RequestId": "d83e0740-e3c6-4e67-b194-2b9081f50e0a"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

获取机架列表

最近更新时间：2025-05-27 01:17:35

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

获取机架列表

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：DescribeRacks。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
Offset	否	Integer	偏移量，默认为0。关于Offset的更进一步介绍请参考 API 简介中的相关小节。 示例值：1
Limit	否	Integer	返回数量，默认为20，最大值为100。关于Limit的更进一步介绍请参考 API 简介中的相关小节。 示例值：5
Filters.N	否	Array of Filter	过滤条件 rack-id 按照机架id进行过滤。 类型：String 必选：否 rack-name 按照机架名称进行过滤。 类型：String 必选：否 idc-id 按照机房id进行过滤。 类型：String 必选：否 idc-unit-id 按照机房管理单元id过滤 类型： String 必选： 否 is-power-on

参数名称	必选	类型	描述
			按照是否开电进行过滤，支持传入 0 和 1。1 代表开电，0 代表关电 类型： String 必选： 否 hosting-type 按照托管类型进行过滤。支持传入 CUSTOMER_MIX 代表客户混合，CUSTOMER_ONLY 代表客户独享 ，NOT_INIT 代表未初始化 类型： String 必选： 否
DstService	否	String	传入目标服务，返回允许进行此服务的机架列表；可以和Filters一起使用。允许的值：('rackPowerOn', 'rackPowerOff') 示例值：rackPowerOn
RackName	否	String	机架名称关键字实现模糊搜索 示例值：R

3. 输出参数

参数名称	类型	描述
RackSet	Array of Rack	客户拥有的机架列表
Total	Integer	总数 示例值：10
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 获取机架列表

输入示例

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

{}
```

输出示例

```
{
"Response": {
"RackSet": [
{
"HostingType": "客户独享",
```

```
"IdcId": 159,
"IdcName": "天津数据备份中心东区DC",
"IdcUnitId": 568,
"IdcUnitName": "天津数据备份中心东区DC1栋M301",
"IsPowerOn": true,
"RackId": 15082,
"RackName": "M301-E10",
"RackOpenTime": "2021-09-06"
}
],
"RequestId": "4df8264f-8348-4e02-98d3-bb4e85c651be",
"Total": 2217
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

获取机位列表

最近更新时间：2025-04-25 01:16:37

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

获取机位列表

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：DescribePositions。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
Offset	否	Integer	偏移量，默认为0。关于Offset的更进一步介绍请参考 API 简介中的相关小节。 示例值：1
Limit	否	Integer	返回数量，默认为20，最大值为100。关于Limit的更进一步介绍请参考 API 简介中的相关小节。 示例值：5
Filters.N	否	Array of Filter	• rack-id 按照【机架ID】进行过滤。例如：15082。 类型：String 必选：否 • rack-name 按照【机架名称】进行过滤，机架名称例如：M301-E10。 类型：String 必选：否 • idc-id 按照【机房ID】进行过滤，机房ID例如：159。 类型：String 必选：否 • idc-unit-id 按照【机房管理单元ID】进行过滤，机房管理ID例如：568。 类型：String 必选：否 • position-status

参数名称	必选	类型	描述
			按照【**机位状态**】进行过滤，机位状态只包含以下四种：机位状态,0 空闲,1 已用,2 不可用,3 预占用,4 预留，例如： 0。 类型：String 必选：否 • **op-status** 按照【**操作状态**】进行过滤，操作状态只包含两种：FINISH 操作完成，PENDING 操作中，例如： PENDING。 类型：String 必选：否

3. 输出参数

参数名称	类型	描述
PositionSet	Array of Position	客户拥有的机位列表
Total	Integer	总数 示例值：10
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 获取机位列表

输入示例

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

{}
```

输出示例

```
{
"Response": {
"PositionSet": [
{
"Height": 2,
"IdcId": 159,
"IdcName": "天津数据备份中心东区DC",
"IdcUnitId": 596,
```

```
"IdcUnitName": "天津数据备份中心东区DC1栋M303",
"PlanDeviceType": 8,
"PositionCode": "10",
"PositionId": 158660,
"PositionStatus": 0,
"RackId": 15451,
"RackName": "M303-C14"
}
],
"RequestId": "718aa630-5ccb-4ffa-983b-f627c433379e",
"Total": 37205
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

获取机房和机房管理单元信息

最近更新时间：2025-04-25 01:16:37

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

获取机房和机房管理单元信息

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：DescribeIdcs。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。

3. 输出参数

参数名称	类型	描述
IdcSet	Array of Idc	机房管理单元列表 示例值：[{"IdcName": "天津数据备份中心东区DC", "IdcId": 555, "IdcUnitSet": [{"IdcUnitId": 568, "IdcUnitName": "天津数据备份中心东区DC1栋M301"}]}]
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 请求示例

输入示例

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

{}
```

输出示例


```
{
"Response": {
"IdcSet": [
{
"IdcId": 373,
"IdcName": "天津数据备份中心东区3栋DC",
"IdcUnitSet": [
{
"CageSet": [
{
"CageName": "cage2",
"CheckerSet": []
}
],
"IdcUnitId": 2555,
"IdcUnitName": "天津数据备份中心东区3栋DCM202"
}
]
}
],
"RequestId": "b8211320-9409-4ec7-b75f-11674b2bc9fc"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

获取设备列表

最近更新时间：2025-04-25 01:16:37

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

获取设备列表

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：DescribeDeviceList。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。
DeviceType	是	String	设备类型 server 服务器，netDevice 网络设备，otherDevice 其他设备 示例值：server
Filters.N	否	Array of Filter	• rack-id 按照【 机架ID 】进行过滤。例如：15082。 类型：String 必选：否 • sn 按照【 设备 SN 码 】进行过滤，设备 SN 例如：TEN948P004。 类型：String 必选：否 • idc-id 按照【 机房ID 】进行过滤，机房ID例如：159。 类型：String 必选：否 • idc-unit-id 按照【 机房管理单元ID 】进行过滤，机房管理ID例如：568。 类型：String 必选：否 • server-type-id 按照【 机器子类型 】进行过滤，只包含以下几种：1:服务器, 2:Twins主机, 3:Twins子机,5:虚拟机, 6:2U4S主机, 7:2U4S子机,8 Rack主机,9 Rack子机，例如： 1。

参数名称	必选	类型	描述
			类型：String 必选：否 • **status** 按照【**设备状态**】进行过滤，操作状态只包含：POWER_ON 设备开电，POWER_OFF 设备关电，RACK_OFF 未上架，MOVING 搬迁中 。例如： POWER_OFF。 类型：String 必选：否 • **svr-is-special** 按照【**是否**】进行过滤，支持 0：自有，1 租用。例如： 1。 类型：String 必选：否
Offset	否	Integer	偏移量，默认为0 示例值：0
Limit	否	Integer	返回数量，默认为20，最大值为1000 示例值：100
DstService	否	String	传入目标服务，返回允许进行此服务的设备列表；可以和Filters一起使用。允许的值：('rackOn', 'powerOn', 'powerOff', 'rackOff', 'quit', 'moving'，'netDeviceCommon', 'serverCommon') 示例值：rackOn

3. 输出参数

参数名称	类型	描述
Total	Integer	总数 示例值：2
DeviceSet	Array of Device	服务器列表
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询服务器设备列表

输入示例

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

```
{
"DeviceType": "server"
}
```

输出示例


```
{
"Response": {
"DeviceSet": [
{
"AssetId": "TH24122400001",
"DeviceType": "server",
"IdcId": 159,
"IdcName": "天津数据备份中心东区DC",
"IdcUnitId": 596,
"IdcUnitName": "天津数据备份中心东区DC1栋M303",
"Ip": "",
"ModelVersion": "DELL R420-V2",
"OnshelfDate": "2025-01-02",
"PositionCode": 20,
"PowerOnTime": "2025-01-02",
"RackId": 15452,
"RackName": "M303-C15",
"ServerTypeId": 1,
"Sn": "20241218chc0001",
"Status": "POWER_OFF",
"SvrIsSpecial": 0
}
],
"RequestId": "13e97f35-2559-4a69-a4ea-38c5d9f42c80",
"Total": 2513
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

查询客户信息

最近更新时间：2025-04-25 01:16:37

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

查询客户信息

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：DescribeCustomerInfo。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。

3. 输出参数

参数名称	类型	描述
CustomerInfo	CustomerInfo	客户信息 示例值： "CustomerInfo": { "CustomerName": "某某科技有限公司", "ShortCustomerName": "某某公司" }
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 获取用户信息

输入示例

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

{}
```

输出示例

```
{
"Response": {
"CustomerInfo": {
"CustomerName": "XX科技有限公司",
"ShortCustomerName": "XX公司",
"WholeFlag": true
},
"RequestId": "5026a8d1-3619-4d79-89e8-0f6e63357a77"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

获取园区列表

最近更新时间：2025-04-25 01:16:38

1. 接口描述

接口请求域名： chc.tencentcloudapi.com 。

获取用户可操作的园区列表

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：DescribeCampusList。
Version	是	String	公共参数，本接口取值：2023-04-18。
Region	是	String	公共参数，详见产品支持的 地域列表。

3. 输出参数

参数名称	类型	描述
CampusSet	Array of Campus	客户可操作的园区列表 示例值： [{'CampusId': 6, 'CampusName': '天津-滨海'}]
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 请求示例

输入示例

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

{}
```

输出示例


```
{
"Response": {
"CampusSet": [
{
"CampusId": 6,
"CampusName": "天津-滨海"
},
{
"CampusId": 163,
"CampusName": "天津-武清"
}
],
"RequestId": "28707dca-e6ba-4e19-97bb-c7c1753865d3"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

数据结构

最近更新时间：2025-05-27 01:17:37

AvailableModelVersion

已通过设备型号评估的型号信息

被如下接口引用：DescribeAvailableModelList。

名称	类型	描述
ModelVersion	String	带有版本号的设备型号 示例值：IBM X3650 M5-V1
DevHeight	String	设备高度 示例值：2
DeviceType	String	设备类型，server 服务器，netDevice 网络设备 示例值：server

Cage

围笼

被如下接口引用：DescribeIdcs。

名称	类型	必选	描述
CageName	String	是	围笼名称 示例值：W1
CheckerSet	Array of String	是	围笼审核人账号ID 示例值：[”700000442974“]

Campus

园区信息

被如下接口引用：DescribeCampusList。

名称	类型	描述
CampusId	Integer	园区ID 示例值：163
CampusName	String	园区名称 示例值：天津-滨海

CommonServiceBaseInfo

通用服务的基本信息

被如下接口引用：DescribeCommonServiceWorkOrderDetail。

名称	类型	描述
IdcName	String	机房名称 示例值：天津数据备份中心东区3栋DC

名称	类型	描述
ContactName	String	业务联系人 示例值：张三
ContactPhone	String	联系人电话 示例值：13800138000
Instructions	String	操作说明 示例值：检查五台设备
ServiceLevel	Integer	1 、2 、3 分别代表 L1、L2、L3 示例值：1
PreAuthorization	Boolean	操作预授权 示例值：true

CustomerInfo

客户信息

被如下接口引用：DescribeCustomerInfo。

名称	类型	描述
CustomerName	String	公司全称 示例值：某某有限责任公司
ShortCustomerName	String	公司简称 示例值：某某公司
WholeFlag	Boolean	是否全托管用户 示例值：true

CustomerReceipt

客户上门自提信息

被如下接口引用：CreateQuitWorkOrder, CreateSpeciallyQuitWorkOrder, DescribeDeviceWorkOrderDetail。

名称	类型	必选	描述
PickUpStuff	String	是	自提人员姓名 示例值：张三
PickUpStuffContact	String	是	自提人电话 示例值：15311111111
PickUpStuffIDCard	String	是	自提人证件号码 示例值：100000000000000005
PickUpTime	String	是	自提时间 示例值：2024-12-11 12:00:00
IDCardType	String	否	证件类型，非必传，默认为IDENTITY_CARD。 对应关系如下：IDENTITY_CARD: 身份证, HONG_KONG_AND_MACAO_PASS: 港澳通行证', PASSPORT: 护照, DRIVING_LICENSE: 驾照,

名称	类型	必选	描述
			OTHER: 其他 示例值：IDENTITY_CARD

Device

服务器信息

被如下接口引用：DescribeDeviceList。

名称	类型	描述
Sn	String	设备 SN 码 示例值：TYSV09090399
ModelVersion	String	设备型号版本 示例值：DELL R410-V1
AssetId	String	设备固资号。只有设备类型为服务器时才返回 示例值：67DGQ2X
SvrIsSpecial	Integer	0 自有，1 租用。只有设备类型为服务器时才返回 示例值：是否租用
Ip	String	IP。 示例值：9.8.174.45
IdcName	String	设备所属的机房名称 示例值：上海腾讯宝信DC
IdcId	Integer	设备所属的机房ID 示例值：122
IdcUnitId	Integer	设备所属的机房管理单元ID 示例值：5587
IdcUnitName	String	设备所属的机房管理单元名称 示例值：上海腾讯宝信DC电信2号楼0203
RackId	Integer	已上架设备所在的机架ID，未上架设备不返回 示例值：1265
ServerTypeId	Integer	服务器类型， 1 代表服务器， 7 代表 2U4S。只有设备类型为服务器时才返回 示例值：1
RackName	String	已上架设备所在的机架名称，未上架设备不返回 示例值：0203-F01
PositionCode	Integer	已上架设备所在的机位编号，未上架设备不返回。只有设备类型为服务器时才返回 示例值：2
Status	String	设备状态：POWER_ON 已开电 POWER_OFF 未开电 RACK_OFF 未上架 MOVING 搬迁中 示例值：POWER_ON
PowerOnTime	String	设备最近一次的开电时间，YYYY-MM-DD 格式。 示例值：2024-11-05
OnshelfDate	String	设备最近一次的上架时间，YYYY-MM-DD 格式。 示例值：2024-11-05
DeviceType	String	设备类型 server 服务器，netDevice 网络设备

名称	类型	描述
		示例值：server
Manufacturer	String	厂商 示例值：华为
TypeName	String	其他设备-设备子类型 示例值：'USB', '移动硬盘', '网络设备板卡', '网络设备模块', '服务器硬盘', '服务器内存', '其他'
HardwareMemo	String	硬件备注 示例值：备注信息

DeviceHistory

工单的设备信息

被如下接口引用：DescribeDeviceWorkOrderDetail。

名称	类型	描述
Sn	String	设备sn 示例值：sn111555
DeviceType	String	设备类型 示例值：server
RackName	String	机架名 示例值：0301-C09
PositionCode	Integer	机位号 示例值：10
IdcId	Integer	机房id 示例值：357
IdcName	String	机房名称 示例值：广州移动华新园AC
IdcUnitId	Integer	机房管理单元id 示例值：2793
IdcUnitName	String	机房管理单元名称 示例值：广州移动华新园AC3楼0301
AssetId	String	固资号 示例值：TH1111111
ModelVersion	String	设备型号-版本，只有收货单详情返回 示例值：Dell R720-V1
DeviceHeight	String	设备高度，只有收货单详情返回 示例值：2
Need10GbSlot	String	需要万兆机位，只有收货单详情返回 示例值：YES
NeedDCPower	String	需要直流电，只有收货单详情返回 示例值：YES
NeedExtranet	String	需要外网，只有收货单详情返回 示例值：NO

名称	类型	描述
NeedVirtualization	String	需要虚拟化，只有收货单详情返回 示例值：NO
Manufacturer	String	厂商,只有收货单详情返回 示例值：华为
HardwareMemo	String	硬件备注 示例值：备注
DstRackName	String	目标机架 示例值：0301-C09
DstPositionCode	String	目标机位 示例值：11
DstIp	String	目标ip 示例值：1.1.1.1
TypeName	String	设备子类型, 其他设备/线材用 示例值：移动硬盘
Quantity	Integer	线材-数量，只有收货单详情返回 示例值：2
Unit	String	计量单位，，只有收货单详情返回，'箱', '卷', '套' 示例值：箱
ReceiptNumber	String	线材-收货凭证号，只有收货单详情返回 示例值：610111111

DeviceOrderBaseInfo

设备类工单的基础历史入参信息

被如下接口引用：DescribeDeviceWorkOrderDetail。

名称	类型	描述
IdcId	Integer	机房id 示例值：691
IdcName	String	机房名称 示例值：广州移动华新园AC
DeviceType	String	设备类型 示例值：server
Remark	String	备注 示例值：备注
ReceivingOperation	String	1.收货-仅核对外包装完整和数量，不开箱 2.验收-开箱核对型号SN一致 示例值：2
EntryTime	String	设备收货-进入时间 示例值：2024-12-12 08:00:00
IsExpressDelivery	Boolean	设备收货-是否快递寄件 示例值：false
ExpressInfo	ExpressDelivery	设备收货-快递寄件信息

名称	类型	描述
StuffOption	String	上/下架人员 1.自行解决 2.由腾讯IDC负责 示例值：1
SelfOperationInfo	SelfOperation	上/下架自行解决信息
WithPowerOn	Boolean	上架后开电 示例值：true
IsPowerOffConfirm	String	关电确认 1.授权时关电 2.关电前需要确认 示例值：1
PowerOffConfirmInfo	PowerOffConfirm	关电前需要确认信息
HandoverMethod	String	退出交接方式 1.物流上门收货 2.客户上门自提 示例值：1
CustomerReceipt	CustomerReceipt	客户上门自提信息
LogisticsReceipt	LogisticsReceipt	物流上门收货信息

DevicePosition

设备及位置信息

被如下接口引用：CreateCommonServiceWorkOrder, DescribeCommonServiceWorkOrderDetail。

名称	类型	必选	描述
Sn	String	是	设备SN 示例值：2174042
RackName	String	是	机架名称 示例值：M303-C15
IdcUnitId	Integer	是	机房管理单元ID 示例值：159
IdcName	String	否	机房名称 示例值：天津数据备份中心东区DC
IdcUnitName	String	否	机房管理单元名称 示例值：天津数据备份中心东区DC1栋M303
AssetId	String	否	设备固资。只有服务器设备才需要这个值 示例值：TYSV17092XXX
PositionCode	Integer	否	机位编号，只有服务器设备才需要传这个值 示例值：20
DeviceType	String	否	server 代表服务器，netDevice 代表网络设备 示例值：server

DeviceRackOn

设备上架信息

被如下接口引用：CreateMovingWorkOrder, CreateRackOnWorkOrder, CreateReceivingWorkOrder。

名称	类型	必选	描述
DeviceSn	String	是	设备sn 示例值：SN123456w-2
DstRackName	String	是	目标机架 示例值：M302-J01
DstPositionCode	String	否	目标机位,服务器必传,网络设备不用传 示例值：2
DstIp	String	否	设备ip 示例值：1.1.1.1

Distribution

机架用量分布

被如下接口引用：DescribeRacksDistribution。

名称	类型	描述
RackNumber	String	机架编号 示例值：C
RackUsageSet	Array of RackUsage	机架的用量分布 示例值：[{ "RackId": 132316, "UsedNum": 1, "UnusedNum": 0, "RackShortName": "C05" }]

ExpressDelivery

快递寄件信息,快递寄件必填

被如下接口引用：CreateReceivingWorkOrder, DescribeDeviceWorkOrderDetail。

名称	类型	必选	描述
LogisticsCompany	String	是	物流公司 示例值：顺丰
ExpressNumber	String	是	快递单号 示例值：100111111

Filter

描述键值对过滤器，用于条件过滤查询。例如过滤ID、名称、状态等

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

被如下接口引用：DescribeDeviceList, DescribeModelVersionList, DescribePositionStatusSummary, DescribePositions, DescribeRacks, DescribeResourceUsage, DescribeWorkOrderList。

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

名称	类型	必选	描述
			示例值：214

Idc

机房信息

被如下接口引用：DescribeIdcs。

名称	类型	描述
IdcName	String	机房名称 示例值：天津数据备份中心东区DC
IdcId	Integer	机房ID 示例值：568
IdcUnitSet	Array of IdcUnit	机房管理单元详情列表 示例值：[{ "IdcUnitId": 568, "IdcUnitName": "天津数据备份中心东区DC1栋M301" }]

IdcUnit

机房管理单元

被如下接口引用：DescribeIdcs。

名称	类型	描述
IdcUnitId	Integer	机房管理单元 ID 示例值：1
IdcUnitName	String	机房管理单元名称 示例值：天津数据备份中心东区DC1栋M301
CageSet	Array of Cage	围笼列表 示例值：[{"CageName": "W1", "CheckerSet": ["700000442974"]}]

IdcUnitInfo

机房管理单元

被如下接口引用：DescribeIdcUnitAssetDetail, DescribeIdcUnitDetail。

名称	类型	描述
Address	String	机房管理单元地址 示例值：山东省济南市经十西路宋庄立交南首担山屯中国联通国家数据中心担山屯南楼二楼南机房
Operator	String	机房经理 示例值：tonyzhang
TelNumber	String	联系电话 示例值：0531-87581111
AssetManager	String	资产管理员 示例值：tonyzhang
AssetManagerTelNumber	String	资产管理员电话 示例值：0531-87581111

LogisticsReceipt

物流上门收货信息

被如下接口引用：CreateQuitWorkOrder, CreateSpeciallyQuitWorkOrder, DescribeDeviceWorkOrderDetail。

名称	类型	必选	描述
LogisticsArrivalTime	String	是	物流预计上门时间 示例值：2024-12-11 12:00:00
LogisticsCompany	String	是	物流公司名称 示例值：顺丰
LogisticsStuff	String	是	物流联系人 示例值：张三
LogisticsStuffContact	String	是	物流电话 示例值：15311111111
ReceiverContact	String	是	收货人电话 示例值：15311111111
ReceiverName	String	是	收货人姓名 示例值：李四
ShippingAddress	String	是	收货地址 示例值：某某地

ModelEvaluationBaseInfo

设备评估工单基本信息

被如下接口引用：DescribeModelEvaluationWorkOrderDetail。

名称	类型	描述
CustomerName	String	客户名称 示例值：腾讯
DeviceType	String	server 服务器 netDevice 网络设备 示例值：设备类型
CampusName	String	园区名称 示例值：天津-武清
Remark	String	备注 示例值：设备入场

ModelVersion

型号以及版本号

被如下接口引用：CreateModelEvaluationWorkOrder。

名称	类型	必选	描述
DevModel	String	是	型号名称 示例值：SFT1502-III
Version	String	是	版本

名称	类型	必选	描述
			示例值：V1

ModelVersionCount

型号和对应的版本数量

被如下接口引用：DescribeModelVersionList。

名称	类型	描述
DevModel	String	型号名称 示例值：ASA 5555
VersionCount	Integer	版本数量 示例值：1

ModelVersionDetail

带有园区评估记录的型号详情

被如下接口引用：DescribeModel, DescribeModelEvaluationWorkOrderDetail。

名称	类型	描述
Version	String	版本号 示例值：V1
CheckResult	Integer	0 代表在当前园区没评估过，1 代表完全满足IDC准入标准 2 代表存在托管风险 3 代表不满足IDC准入标准 示例值：0
OptionSet	Array of TemplateOption	型号各个配置项的详情
ModelVersion	String	设备型号名称及版本 示例值：IBM X3650 M5-V1

NetDeviceModel

网络设备型号详情

被如下接口引用：CreateNetDeviceModel。

名称	类型	必选	描述
Version	String	否	版本号 示例值：V1
ModelVersion	String	否	型号和版本的组合名称 示例值：Cisco IronPort 370-V1
DevModel	String	否	型号名 示例值：Cisco IronPort 370
DevWidth	String	否	宽度 示例值：370－446

名称	类型	必选	描述
DevDepth	String	否	深度 示例值：≤780
DevWeight	String	否	重量 示例值：≤50
MountEar	String	否	是否携带挂耳 示例值：是
AccordCCC	String	否	是否符合CCC认证 示例值：是
PassNetwork	String	否	是否通过入网许可认证 示例值：是
PowerportType	String	否	电源接口型号 示例值：C13
PowerModule	String	否	电源模块 示例值：直流-270DC
PowermoduleNum	String	否	电源模块数量 示例值：2
PowermodulePosition	String	否	电源模块位置 示例值：设备后端
HighVoltageAdapt	String	否	高压直流自适应 示例值：是
PowerEnergy	String	否	实际工作功耗(W) 示例值：400
InwindPosition	String	否	进风口位置 示例值：前
OutwindPosition	String	否	出风口位置 示例值：后
BusinessPortPosition	String	否	业务端口位置 示例值：前
LineManager	String	否	带有理线器 示例值：是
CheckResult	Integer	否	0 代表在当前园区没评估过，1 代表完全满足IDC准入标准 2 代表存在托管风险 3 代表不满足IDC准入标准 示例值：0
DevHeight	String	否	设备高度 示例值：2

NetReceivingInfo

网络设备收货详情

被如下接口引用：CreateReceivingWorkOrder。

名称	类型	必选	描述
DeviceSn	String	是	设备sn 示例值：TH55555
ModelVersion	String	是	设备型号-版本 示例值：HUAWEI RH2288-V1
HardwareMemo	String	否	硬件备注 示例值：备注
Manufacturer	String	否	厂商 示例值：华为

OptionValueItem

型号选项下拉框中的选项值

被如下接口引用：DescribeModelTemplate。

名称	类型	必选	描述
OptionValue	String	是	选项的值 示例值：1
Selected	Boolean	是	是否默认选中 示例值：true

OrderStep

工单详情中的工单流程步骤

被如下接口引用：DescribeCommonServiceWorkOrderDetail, DescribeDeviceWorkOrderDetail, DescribeModelEvaluationWorkOrderDetail, DescribePersonnelVisitWorkOrderDetail。

名称	类型	描述
StepName	String	步骤名 示例值：数经审核
OwnerName	String	处理人 示例值：张三
OwnerPhone	String	处理人手机号 示例值：15311111111
FinishTime	String	完成时间 示例值：2024-12-12 08:00:00
StepStatus	String	此步骤状态 示例值：finish

OtherDevReceivingInfo

其他设备收货信息

被如下接口引用：CreateReceivingWorkOrder, CreateSpeciallyQuitWorkOrder。

名称	类型	必选	描述
DeviceSn	String	是	设备sn 示例值：td11111
TypeName	String	是	'USB', '移动硬盘', '网络设备板卡', '网络设备模块', '服务器硬盘', '服务器内存', '其他' 示例值：移动硬盘
Manufacturer	String	否	厂商 示例值：华为
HardwareMemo	String	否	硬件备注 示例值：备注

Personnel

到访人员

被如下接口引用：CreatePersonnelVisitWorkOrder, DescribePersonnelVisitWorkOrderDetail。

名称	类型	必选	描述
IDCardNumber	String	是	证件号码 示例值：123456789012345678
IDCardType	String	是	证件类型。对应关系如下：IDENTITY_CARD: 身份证, HONG_KONG_AND_MACAO_PASS: 港澳通行证', PASSPORT: 护照, DRIVING_LICENSE: 驾照, OTHER: 其他 示例值：IDENTITY_CARD
Company	String	是	公司名称 示例值：某某科技有限公司
LanguageType	String	是	语言。对应关系：ENGLISH: 英文, CHINESE: 中文 示例值：ENGLISH
Name	String	是	姓名 示例值：张三
TelNumber	String	是	电话 示例值：13800138000
Position	String	否	职位 示例值：工程师
Wechat	String	否	微信 示例值：zhangsan_wechat
Email	String	否	邮箱 示例值：zhangsan@example.com

PersonnelVisitBaseInfo

人员到访工单基本信息

被如下接口引用：DescribePersonnelVisitWorkOrderDetail。

名称	类型	描述
IdcName	String	机房名称 示例值：天津数据备份中心东区DC
VisitReason	Array of String	到访原因。到访原因，映射关系：DEVICE_MAINTENANCE 设备维护 DEVICE_MOVE 设备收货上下架 CHECK 盘点 OTHER 其他 示例值：设备类型
VisitRemark	String	到访原因 示例值：检查设备
EnterStartTime	String	到访结束时间 示例值：2024-10-27 00:25:56
EnterEndTime	String	到访开始时间 示例值：2024-10-27 00:25:56
IdcUnitNameList	Array of String	机房管理单元列表 示例值：["天津数据备份中心东区DC1栋M301", "天津数据备份中心东区DC1栋M303"]

Position

机位信息

被如下接口引用：DescribePositions。

名称	类型	描述
PositionId	Integer	机位ID 示例值：1787308
Height	Integer	机位高度 示例值：2
PositionCode	String	机位编号 示例值：2
PositionStatus	Integer	机位状态,0 空闲,1 已用,2 不可用,3 预占用,4 预留 示例值：1
PlanDeviceType	Integer	设备规划类型ID 示例值：1
IdcUnitId	Integer	机位所属的机房管理单元ID 示例值：1
RackId	Integer	机位所属的机架ID 示例值：1321
RackName	String	机位所属的机架名称 示例值：2D-C05
IdcUnitName	String	机位所属的机房管理单元名称 示例值：上海腾讯宝信DC移动3号楼2D
IdcName	String	机位所属的机房名称 示例值：上海腾讯宝信DC
IdcId	Integer	机位所属的机房ID 示例值：106

名称	类型	描述
Sn	String	机位上如果有设备，该字段代表设备的 SN 码，如果是空闲机位，不返回该字段。 示例值：SN123456w-2
AssetId	String	机位上如果有设备，该字段代表设备的固资号，如果是空闲机位，不返回该字段。 示例值：TYSV09090338
ModelVersion	String	机位上如果有设备，该字段代表设备的设备型号加版本号，如果是空闲机位，不返回该字段。 示例值：DELL R410-V1

PositionStatusItem

机位状态及对应的数量

被如下接口引用：DescribePositionStatusSummary。

名称	类型	描述
PositionStatus	Integer	状态值 示例值：0
Count	Integer	对应的机位数量 示例值：10

PowerOffConfirm

关电确认信息

被如下接口引用：CreatePowerOffWorkOrder, CreateQuitWorkOrder, CreateRackOffWorkOrder, DescribeDeviceWorkOrderDetail。

名称	类型	必选	描述
ConfirmContact	String	是	联系人 示例值：张三
ConfirmContactNumber	String	是	联系人电话 示例值：15311111111

Rack

机架的信息

被如下接口引用：DescribeRacks。

名称	类型	描述
RackName	String	机架名称 示例值：M301-E10
IdcUnitId	Integer	机架所属的机房管理单元ID 示例值：8
IdcUnitName	String	机架所属的机房管理单元名称 示例值：北京腾讯银C16楼
IdcName	String	机架所属的机房名称 示例值：北京腾讯银科大厦ODC

名称	类型	描述
IdcId	Integer	机架所属的机房ID 示例值：106
RackId	Integer	机架ID 示例值：1211
IsPowerOn	Boolean	是否开电 示例值：true
RackOpenTime	String	机架最近一次开电时间，YYYY-MM-DD 格式 示例值：2008-10-31
HostingType	String	托管类型 示例值：客户独享

RackUsage

机架用量

被如下接口引用：DescribeRacksDistribution。

名称	类型	描述
RackId	Integer	机架ID 示例值：132316
UsedNum	Integer	已使用的机位数量 示例值：1
UnusedNum	Integer	空闲机位数量 示例值：0
RackShortName	String	机架简称 示例值：C05
TotalNum	Integer	机位总数 示例值：1
UsedRate	Float	机位使用率 示例值：0.29

SelfOperation

客户自行上门信息

被如下接口引用：CreateQuitWorkOrder, CreateRackOffWorkOrder, CreateRackOnWorkOrder, CreateReceivingWorkOrder, DescribeDeviceWorkOrderDetail。

名称	类型	必选	描述
StuffContact	String	是	联系人员电话 示例值：15311111111
StuffIDCard	String	是	证件号码 示例值：100000000000000005
StuffName	String	是	人员姓名 示例值：张三

名称	类型	必选	描述
OperationTime	String	是	上门时间 示例值：2024-11-20 12:00:00
IDCardType	String	否	证件类型，非必传，默认为IDENTITY_CARD。 对应关系如下：IDENTITY_CARD: 身份证, HONG_KONG_AND_MACAO_PASS: 港澳通行证', PASSPORT: 护照, DRIVING_LICENSE: 驾照, OTHER: 其他 示例值：IDENTITY_CARD

ServerModel

服务器设备型号

被如下接口引用：CreateServerModel。

名称	类型	必选	描述
DevModel	String	否	型号名称 示例值：SFT1502-III
DevNode	String	否	节点数 示例值：1
DevHeight	String	否	设备高度 示例值：2
PowerEnergy	String	否	功耗 示例值：500
PowerportType	String	否	电源接口型号 示例值：C14
PowermoduleNum	String	否	电源模块数量 示例值：2
InwindPosition	String	否	进风口位置 示例值：前
OutwindPosition	String	否	出风口位置 示例值：后
NetportPosition	String	否	网卡接口位置 示例值：设备后端
DevWidth	String	否	宽度 示例值：430
DevDepth	String	否	深度 示例值：500
DevWeight	String	否	重量 示例值：18
PowerModule	String	否	电源模块 示例值：兼容直流270DC和交流
PowermodulePosition	String	否	电源模块位置 示例值：设备后端

名称	类型	必选	描述
NetportType	String	否	网络接口类型 示例值：电口
NetSpeed	String	否	网卡速率 示例值：千兆
CheckResult	Integer	否	0 代表在当前园区没评估过，1 代表完全满足IDC准入标准 2 代表存在托管风险 3 代表不满足IDC准入标准 示例值：0
Version	String	否	版本号 示例值：V1
ModelVersion	String	否	型号和版本的组合名称 示例值：Cisco IronPort 370-V1

ServerReceivingInfo

服务器收货信息

被如下接口引用：CreateReceivingWorkOrder。

名称	类型	必选	描述
DeviceSn	String	是	设备sn 示例值：TH55555
ModelVersion	String	是	设备型号-版本 示例值：HUAWEI RH2288-V1
Need10GbSlot	String	否	需要万兆机位 示例值：NO
NeedDCPower	String	否	需要直流电 示例值：NO
NeedExtranet	String	否	需要外网 示例值：YES
NeedVirtualization	String	否	需要虚拟化 示例值：YES
HardwareMemo	String	否	硬件备注 示例值：备注

TemplateOption

型号模板的选项

被如下接口引用：DescribeModel, DescribeModelEvaluationWorkOrderDetail, DescribeModelTemplate。

名称	类型	描述
OptionName	String	选项名称 示例值：高度(u)
Standard	String	腾讯的标准 示例值：40

名称	类型	描述
StandardInfo	String	腾讯标准的展示字段 示例值：≤40u(1u=44.45mm)
OptionKey	String	选项的唯一标识 示例值：devHeight
InputType	String	输入的类型 示例值：int
ValueType	String	输入值的类型 示例值：number
CompareType	String	是否符合腾讯标准的比较方式，-- 为不做比较 示例值：<=
OptionValueSet	Array of OptionValueItem	下拉列表中填充的选项值 示例值：[{"OptionValue": "1", "Selected": true}]
InputHint	String	输入框中的占位的提示符 示例值：填写U数
InputInfo	String	输入框下方的提示文案 示例值：请选择
OptionValue	String	客户写入的值 示例值：C14

WireReceivingInfo

线材收货信息

被如下接口引用：CreateReceivingWorkOrder。

名称	类型	必选	描述
TypeName	String	是	'光纤', '网线', '电源线' 示例值：光纤
Quantity	Integer	是	数量 示例值：2
Unit	String	是	'箱', '卷', '套' 示例值：箱
ReceiptNumber	String	是	收货凭证号 示例值：10000001
HardwareMemo	String	否	硬件备注 示例值：备注

WorkOrderData

工单的常用信息返回

被如下接口引用：DescribeWorkOrderList。

名称	类型	描述
WorkOrderId	String	工单号 示例值：ord-24111418091131849
ServiceType	String	服务类型，一个服务可能会产生多个工单 示例值：moving
OrderType	String	工单类型 示例值：rackOn
OrderStatus	String	工单状态 示例值：processing
Creator	String	工单创建人 示例值：tornado
CreateTime	String	工单创建时间 示例值：2024-11-14 18:09:11
FinishTime	String	工单完成时间 示例值：2024-11-15 20:04:11

WorkOrderFamilyDetail

带有分类的工单类型列表

被如下接口引用：DescribeWorkOrderTypes。

名称	类型	描述
WorkOrderFamily	String	工单类型大类的名称 示例值：设备服务
WorkOrderTypeSet	Array of WorkOrderTypeDetail	工单类型详情列表

WorkOrderTinyInfo

工单信息的简要，一般用于工单创建的返回

被如下接口引用：CreateCommonServiceWorkOrder, CreateModelEvaluationWorkOrder, CreateMovingWorkOrder, CreatePersonnelVisitWorkOrder, CreatePowerOffWorkOrder, CreatePowerOnWorkOrder, CreateQuitWorkOrder, CreateRackOffWorkOrder, CreateRackOnWorkOrder, CreateReceivingWorkOrder, CreateSpeciallyQuitWorkOrder。

名称	类型	描述
WorkOrderId	String	工单id 示例值：ord-24111418091171613
ServiceType	String	服务类型，一个服务可能会产生多个工单 示例值：powerOn
OrderType	String	工单类型 示例值：rackOff

WorkOrderTypeDetail

工单类型详情

被如下接口引用：DescribeWorkOrderTypes。

名称	类型	描述
WorkOrderFamily	String	工单类型所属的大类 示例值：设备服务
WorkOrderName	String	工单类型名称 示例值：设备型号评估
WorkOrderType	String	工单类型的唯一英文标识 示例值：modelEvaluation
WorkOrderDescription	String	工单类型简述 示例值：客户设备托管到腾讯IDC机房，为了确保设备符合机房的上架标准， 在入场前，需要提前进行托管设备入场的评估。
CollectFlag	Boolean	是否被收藏 示例值：true
SlaMessage	String	服务时效 示例值：2 个工作日

错误码

最近更新时间：2025-06-06 01:18:12

功能说明

如果返回结果中存在 Error 字段，则表示调用 API 接口失败。例如：

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

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

错误码列表

公共错误码

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

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

业务错误码

错误码	说明
InvalidParameterValue.EndTimeLowerThanStartTime	结束时间必须大于起始时间
InvalidParameterValue.InvalidTimeFormat	时间格式不符合规范
InvalidParameterValue.InvalidWorkOrderType	非法的工单类型