软件成分分析

API 文档

腾讯云

文档目录

API 文档
更新历史

最近更新时间：2024-10-29 21:02:29

第 7 次发布

发布时间：2024-10-29 21:02:24

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- DescribeKBComponentVersionList
  - 新增入参：PageNumber, PageSize, Order, OrderBy, Filter
- DescribeKBComponentVulnerability
  - 新增出参：PURL, RecommendedVersion

新增数据结构：

- ComponentTagFilter
- ComponentVersionInfo

修改数据结构：

- Component
  - 新增成员：VersionInfo, LastUpdateTime, TagList
- ComponentVersion
  - 新增成员：VersionInfo
- VulnerabilityDetail
  - 新增成员：UpdateTime
- VulnerabilitySummary
  - 新增成员：Architecture, ArchitectureList, PatchUrlList

第 6 次发布

发布时间：2024-04-16 01:05:23

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- DescribeKBComponentVulnerability
  - 新增入参：Language
- DescribeKBVulnerability
  - 新增入参：Language

第 5 次发布

发布时间：2023-10-24 01:07:40

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- DescribeKBVulnerability
  - 新增入参：CNVDID, CNNVDID

新增数据结构：

- AffectedComponent

修改数据结构：

- VulnerabilityDetail
  - 新增成员：AffectedComponentList

第 4 次发布

发布时间：2023-10-17 01:08:28

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DescribeKBComponentVersionList
- SearchKBComponent

新增数据结构：

- ComponentVersion

第 3 次发布

发布时间：2022-09-29 06:12:03

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- MatchKBPURLList
  - 新增出参：Hit

第 2 次发布

发布时间：2022-06-09 06:06:39

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- MatchKBPURLList

第 1 次发布

发布时间：2022-04-02 17:37:22

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DescribeKBComponent
- DescribeKBComponentVulnerability
- DescribeKBLicense
- DescribeKBVulnerability

新增数据结构：

- CVSSV2Info
- CVSSV3Info
- Component
- ComponentVulnerabilitySummary
- ComponentVulnerabilityUnion
- LicenseDetail
- LicenseRestriction
- LicenseSummary
- LicenseUnion
- PURL
- Qualifier
- VulnerabilityDetail
- VulnerabilitySummary
- VulnerabilityUnion

简介

最近更新时间：2024-10-29 21:02:28

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

软件成分分析（T-Sec-BSCA）是一款基于二进制文件分析能力，对可执行二进制、固件、APK、镜像等格式做自动化软件成分分析的工具。T-Sec-BSCA无需源码，聚焦于漏洞扫描、开源协议审计和敏感信息检测，一键上传即输出安全报告。

API 概览

最近更新时间：2024-10-29 21:02:28

知识库相关接口

接口名称	接口功能	频率限制（次/秒）
DescribeKBComponent	查询知识库组件信息	-
DescribeKBComponentVersionList	查询组件的版本列表	10000
DescribeKBComponentVulnerability	查询知识库组件漏洞	-
DescribeKBLicense	查询知识库许可证信息	-
DescribeKBVulnerability	查询知识库漏洞详情列表	-
MatchKBPURLList	匹配知识库组件列表	10000
SearchKBComponent	搜索组件	10000

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：2024-10-29 21:02:28

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 bsca.tencentcloudapi.com ，也支持指定地域域名访问，例如广州地域的域名为 bsca.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 bsca.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）	bsca.tencentcloudapi.com
华南地区(广州)	bsca.ap-guangzhou.tencentcloudapi.com
华东地区(上海)	bsca.ap-shanghai.tencentcloudapi.com
华北地区(北京)	bsca.ap-beijing.tencentcloudapi.com
西南地区(成都)	bsca.ap-chengdu.tencentcloudapi.com
西南地区(重庆)	bsca.ap-chongqing.tencentcloudapi.com
港澳台地区(中国香港)	bsca.ap-hongkong.tencentcloudapi.com
亚太东南(新加坡)	bsca.ap-singapore.tencentcloudapi.com
亚太东南(曼谷)	bsca.ap-bangkok.tencentcloudapi.com
亚太南部(孟买)	bsca.ap-mumbai.tencentcloudapi.com
亚太东北(首尔)	bsca.ap-seoul.tencentcloudapi.com
亚太东北(东京)	bsca.ap-tokyo.tencentcloudapi.com
美国东部(弗吉尼亚)	bsca.na-ashburn.tencentcloudapi.com
美国西部(硅谷)	bsca.na-siliconvalley.tencentcloudapi.com
欧洲地区(法兰克福)	bsca.eu-frankfurt.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：2023-12-28 01:07:57

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

公共参数的具体内容会因您使用的签名方法版本不同而有所差异。

使用签名方法 v3 的公共参数

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

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

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

参数名称	类型	必选	描述
Action	String	是	HTTP 请求头：X-TC-Action。操作的接口名称。取值参考接口文档输入参数章节关于公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。
Region	String	-	HTTP 请求头：X-TC-Region。地域参数，用来标识希望操作哪个地域的数据。取值参考接口文档中输入参数章节关于公共参数 Region 的说明。**注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。**
Timestamp	Integer	是	HTTP 请求头：X-TC-Timestamp。当前 UNIX 时间戳，可记录发起 API 请求的时间。例如 1529223702。**注意：如果与服务器时间相差超过5分钟，会引起签名过期错误。**
Version	String	是	HTTP 请求头：X-TC-Version。操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
Authorization	String	是	HTTP 标准身份认证头部字段，例如： TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/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，AKIDEXAMPLE 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 bsca；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********EXAMPLE/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********EXAMPLE/2018-05-30/cvm/tc3_request, SignedHeaders=content-type;ho
st, 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********EXAMPLE/2018-05-30/cvm/tc3_request, SignedHeaders=content-type;ho
st, 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。

参数名称	类型	必选	描述
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********EXAMPLE

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********EXAMPLE
```

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：2023-12-27 01:08:29

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：`AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******` 和 `Gu5t9xGARNpq86cd98joQYCN3*******`。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=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host;x-tc-action, Signature=be4f67d323c78ab9acb7395e43c0dbcf822a9cfac32fea2449a7bc7726b770a3" \
-H "Content-Type: application/json; charset=utf-8" \
-H "Host: cvm.tencentcloudapi.com" \
-H "X-TC-Action: DescribeInstances" \
-H "X-TC-Timestamp: 1551113065" \
-H "X-TC-Version: 2017-03-12" \
-H "X-TC-Region: ap-guangzhou" \
-d '{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}'
```

下面详细解释签名计算过程。

1. 拼接规范请求串

按如下伪代码格式拼接规范请求串（CanonicalRequest）：

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

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

字段名称	解释
HashedRequestPayload	请求正文（payload，即 body，此示例为 `{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}`）的哈希值，计算伪代码为 Lowercase(HexEncode(Hash.SHA256(RequestPayload)))，即对 HTTP 请求正文做 SHA256 哈希，然后十六进制编码，最后编码串转换成小写字母。对于 GET 请求，RequestPayload 固定为空字符串。此示例计算结果是 `35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064`。

根据以上规则，示例中得到的规范请求串如下：

```
POST
/

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

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

2. 拼接待签名字符串

按如下格式拼接待签名字符串：

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

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

注意：

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

根据以上规则，示例中得到的待签名字符串如下：

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

3. 计算签名

1）计算派生签名密钥，伪代码如下：

```
SecretKey = "Gu5t9xGARNpq86cd98joQYCN3*******"
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：
f1cb4d518a0eda9d5cbbfdb7850983f1e603eeae484edea76e4dd8d8deb5556e，
e7c609ce81bea53546bed2cc904778bef9ca14082e48e67883443ed64e227cd7，
8aa8ab5755582f576e94bcfe383b8e29325b0ca90c3590d569221c6a63a091ed。

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 `Gu5t9xGARNpq86cd98joQYCN3*******`。
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 `be4f67d323c78ab9acb7395e43c0dbcf822a9cfac32fea2449a7bc7726b770a3` 。

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 `AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******`。
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 `be4f67d323c78ab9acb7395e43c0dbcf822a9cfac32fea2449a7bc7726b770a3`。

根据以上规则，示例中得到的值为：

```
TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host;x-tc-action, Signature=be4f67d323c78ab9acb7395e43c0dbcf822a9cfac32fea2449a7bc7726b770a3
```

最终完整的调用信息如下：

```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host;x-tc-action, Signature=be4f67d323c78ab9acb7395e43c0dbcf822a9cfac32fea2449a7bc7726b770a3
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 AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******
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 Gu5t9xGARNpq86cd98joQYCN3*******
private final static String SECRET_KEY = System.getenv("TENCENTCLOUD_SECRET_KEY");
private final static String CT_JSON = "application/json; charset=utf-8";

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

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

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

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

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

// ************* 步骤 2：拼接待签名字符串 *************
String credentialScope = date + "/" + service + "/" + "tc3_request";
```

```
String hashedCanonicalRequest = sha256Hex(canonicalRequest);
String stringToSign = algorithm + "\n" + timestamp + "\n" + credentialScope + "\n" + hashedCanonicalRequest;
System.out.println(stringToSign);

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

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

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

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

Python

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

# 密钥参数
# 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******
secret_id = os.environ.get("TENCENTCLOUD_SECRET_ID")
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 Gu5t9xGARNpq86cd98joQYCN3*******
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 AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******
secretId := os.Getenv("TENCENTCLOUD_SECRET_ID")
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 Gu5t9xGARNpq86cd98joQYCN3*******
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 AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******
$secretId = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 Gu5t9xGARNpq86cd98joQYCN3*******
$secretKey = getenv("TENCENTCLOUD_SECRET_KEY");
$host = "cvm.tencentcloudapi.com";
```

```php
$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 AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******
secret_id = ENV["TENCENTCLOUD_SECRET_ID"]
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 Gu5t9xGARNpq86cd98joQYCN3*******
secret_key = ENV["TENCENTCLOUD_SECRET_KEY"]

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

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

```
hashed_request_payload,
].join("\n")

puts canonical_request

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

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

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

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

DotNet

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

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

```
{
builder.Append(hashbytes[i].ToString("x2"));
}
return builder.ToString();
}
}

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

public static Dictionary<String, String> BuildHeaders(string secretid,
string secretkey, string service, string endpoint, string region,
string action, string version, DateTime date, string requestPayload)
{
string datestr = date.ToString("yyyy-MM-dd");
DateTime startTime = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc);
long requestTimestamp = (long)Math.Round((date - startTime).TotalMilliseconds, MidpointRounding.AwayFromZero) / 1000;
// ************* 步骤 1：拼接规范请求串 *************
string algorithm = "TC3-HMAC-SHA256";
string httpRequestMethod = "POST";
string canonicalUri = "/";
string canonicalQueryString = "";
string contentType = "application/json";
string canonicalHeaders = "content-type:" + contentType + "; charset=utf-8\n"
+ "host:" + endpoint + "\n"
+ "x-tc-action:" + action.ToLower() + "\n";
string signedHeaders = "content-type;host;x-tc-action";
string hashedRequestPayload = SHA256Hex(requestPayload);
string canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload;
Console.WriteLine(canonicalRequest);

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

// ************* 步骤 3：计算签名 *************
byte[] tc3SecretKey = Encoding.UTF8.GetBytes("TC3" + secretkey);
byte[] secretDate = HmacSHA256(tc3SecretKey, Encoding.UTF8.GetBytes(datestr));
byte[] secretService = HmacSHA256(secretDate, Encoding.UTF8.GetBytes(service));
byte[] secretSigning = HmacSHA256(secretService, Encoding.UTF8.GetBytes("tc3_request"));
```

```
byte[] signatureBytes = HmacSHA256(secretSigning, Encoding.UTF8.GetBytes(stringToSign));
string signature = BitConverter.ToString(signatureBytes).Replace("-", "").ToLower();
Console.WriteLine(signature);

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

Dictionary<string, string> headers = new Dictionary<string, string>();
headers.Add("Authorization", authorization);
headers.Add("Host", endpoint);
headers.Add("Content-Type", contentType + "; charset=utf-8");
headers.Add("X-TC-Timestamp", requestTimestamp.ToString());
headers.Add("X-TC-Version", version);
headers.Add("X-TC-Action", action);
headers.Add("X-TC-Region", region);
return headers;
}
public static void Main(string[] args)
{
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******
string SECRET_ID = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 Gu5t9xGARNpq86cd98joQYCN3*******
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-na
me\"}]}";

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 AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******
const SECRET_ID = process.env.TENCENTCLOUD_SECRET_ID
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 Gu5t9xGARNpq86cd98joQYCN3*******
const SECRET_KEY = process.env.TENCENTCLOUD_SECRET_KEY

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

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

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

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

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

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

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

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

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

C++

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

using namespace std;
```

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

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

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

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

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

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

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

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

return (ss.str());
}

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

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

int main()
{
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******
string SECRET_ID = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 Gu5t9xGARNpq86cd98joQYCN3*******
string SECRET_KEY = getenv("TENCENTCLOUD_SECRET_KEY");

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

// ************* 步骤 1：拼接规范请求串 *************
string httpRequestMethod = "POST";
string canonicalUri = "/";
string canonicalQueryString = "";
string lower = action;
std::transform(action.begin(), action.end(), lower.begin(), ::tolower);
string canonicalHeaders = string("content-type:application/json; charset=utf-8\n")
```

```
+ "host:" + host + "\n"
+ "x-tc-action:" + lower + "\n";
string signedHeaders = "content-type;host;x-tc-action";
string payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name
\"}]}";
string hashedRequestPayload = sha256Hex(payload);
string canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload;
cout << canonicalRequest << endl;

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

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

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

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

C

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

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

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

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

void hmac_sha256(char* key, const char* input, char* result)
{
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, strlen(key), EVP_sha256(), NULL);
HMAC_Update(h, ( unsigned char* )input, strlen(input));
unsigned int len = 32;
HMAC_Final(h, hash, &len);

#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX_cleanup(h);
#else
HMAC_CTX_free(h);
#endif
strncpy(result, (const char*)hash, len);
}

void hex_encode(const char* input, char* output)
{
```

```
static const char* const lut = "0123456789abcdef";
size_t len = strlen(input);
char add_out[128] = {0};
char temp[2] = {0};
for (size_t i = 0; i < 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 AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******
const char* SECRET_ID = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 Gu5t9xGARNpq86cd98joQYCN3*******
const char* SECRET_KEY = getenv("TENCENTCLOUD_SECRET_KEY");
const char* service = "cvm";
const char* host = "cvm.tencentcloudapi.com";
const char* region = "ap-guangzhou";
const char* action = "DescribeInstances";
const char* version = "2017-03-12";
int64_t timestamp = 1551113065;
char date[20] = {0};
get_utc_date(timestamp, date, sizeof(date));

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

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


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

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

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

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

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

```
printf("%s\n", curlcmd);
return 0;
}
```

其他语言

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

签名失败

存在以下签名失败的错误码，请根据实际情况处理。

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

签名方法

最近更新时间：2024-01-05 01:07:55

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: `AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******`
- SecretKey: `Gu5t9xGARNpq86cd98joQYCN3*******`

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	`AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******`
Timestamp	当前时间戳	1465185768
Nonce	随机正整数	11886
Region	实例所在区域	ap-guangzhou
InstanceIds.0	待查询的实例 ID	ins-09dx96dg
Offset	偏移量	0
Limit	最大允许输出	20

参数名称	中文	参数值
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' : 'AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******',
'Timestamp' : 1465185768,
'Version': '2017-03-12',
}
```

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

2.2. 拼接请求字符串

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

然后将格式化后的各个参数用"&"拼接在一起，最终生成的请求字符串为:

```
Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******&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=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******&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 = 'Gu5t9xGARNpq86cd98joQYCN3*******';
$srcStr = 'GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******&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：

```
zmmjn35mikh6pM3V7sUEuX4wyYM=
```

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 zmmjn35mikh6pM3V7sUEuX4wyYM= ，最终得到的签名串请求参数（Signature）为：zmmjn35mikh6pM3V7sUEuX4wyYM%3D，它将用于生成最终的请求 URL。

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

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

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

4. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理。

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

5. 签名演示

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

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

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

- Signature Demo

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

最终输出的 url 可能为：`https://cvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******&Signature=zmmjn35mikh6pM3V7sUEuX4wyYM%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 AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******
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 Gu5t9xGARNpq86cd98joQYCN3******
params.put("Signature", sign(getStringToSign(params), System.getenv("TENCENTCLOUD_SECRET_KEY"), "HmacSHA1")); // 公共参
数
System.out.println(getUrl(params));
}
}
```

Python

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

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

import requests

# 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******
secret_id = os.environ.get("TENCENTCLOUD_SECRET_ID")
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 Gu5t9xGARNpq86cd98joQYCN3*******
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 AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******
secretId := os.Getenv("TENCENTCLOUD_SECRET_ID")
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 Gu5t9xGARNpq86cd98joQYCN3*******
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 AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******
$secretId = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 Gu5t9xGARNpq86cd98joQYCN3*******
$secretKey = getenv("TENCENTCLOUD_SECRET_KEY");
$param["Nonce"] = 11886;//rand();
$param["Timestamp"] = 1465185768;//time();
$param["Region"] = "ap-guangzhou";
$param["SecretId"] = $secretId;
$param["Version"] = "2017-03-12";
$param["Action"] = "DescribeInstances";
$param["InstanceIds.0"] = "ins-09dx96dg";
$param["Limit"] = 20;
$param["Offset"] = 0;

ksort($param);

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

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

```
// curl_close($ch);
// echo json_decode($output);
```

Ruby

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

# 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******
secret_id = ENV["TENCENTCLOUD_SECRET_ID"]
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 Gu5t9xGARNpq86cd98joQYCN3*******
secret_key = ENV["TENCENTCLOUD_SECRET_KEY"]

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

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

DotNet

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

```
public class Application {
public static string Sign(string signKey, string secret)
{
string signRet = string.Empty;
using (HMACSHA1 mac = new HMACSHA1(Encoding.UTF8.GetBytes(signKey)))
{
byte[] hash = mac.ComputeHash(Encoding.UTF8.GetBytes(secret));
signRet = Convert.ToBase64String(hash);
}
return signRet;
}
public static string MakeSignPlainText(SortedDictionary<string, string> requestParams, string requestMethod, string req
uestHost, 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 AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******
string SECRET_ID = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 Gu5t9xGARNpq86cd98joQYCN3*******
string SECRET_KEY = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_KEY");

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

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

param.Add("SecretId", SECRET_ID);
```

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

NodeJS

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

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

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

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

function main(){
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******
const SECRET_ID = process.env.TENCENTCLOUD_SECRET_ID
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 Gu5t9xGARNpq86cd98joQYCN3*******
const SECRET_KEY = process.env.TENCENTCLOUD_SECRET_KEY

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

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

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

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

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

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

返回结果

最近更新时间：2024-03-12 01:11:53

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

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

正确返回结果

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

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

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

错误返回结果

若调用失败，其返回值示例如下为：

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

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

公共错误码

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

参数类型

最近更新时间：2022-08-10 06:16:16

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。

知识库相关接口
匹配知识库组件列表

最近更新时间：2024-03-12 01:11:52

1. 接口描述

接口请求域名： bsca.tencentcloudapi.com 。

本接口(MatchKBPURLList)用于在知识库中匹配与特征对应的开源组件列表。

默认接口请求频率限制：10000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：MatchKBPURLList。
Version	是	String	公共参数，本接口取值：2021-08-11。
Region	否	String	公共参数，本接口不需要传递此参数。
SHA1	否	String	SHA1。 示例值：7ed845de1dfe070d43511fab321784e6c4118398

3. 输出参数

参数名称	类型	描述
PURLList	Array of PURL	组件列表。
Hit	Boolean	是否命中数据库。 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 查询知识库组件列表

输入示例

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

{
```

```
"SHA1": "7ed845de1dfe070d43511fab321784e6c4118398"
}
```

输出示例

```
{
"Response": {
"Hit": true,
"PURLList": [
{
"Name": "log4j-core",
"Namespace": "org.apache.logging.log4j",
"Protocol": "maven",
"Qualifiers": [],
"Subpath": "",
"Version": "2.5"
}
],
"RequestId": "3189252f-aad2-44b2-b11c-b51fccd8bf6d"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.AccountNotEnough	账户流量余额不足。
InternalError	内部错误。
InvalidParameter	参数错误。

错误码	描述
InvalidParameterValue	参数取值错误。
ResourceNotFound	资源不存在。

搜索组件

最近更新时间：2024-10-29 21:02:26

1. 接口描述

接口请求域名： bsca.tencentcloudapi.com 。

根据输入的组件名、组件类型搜索相应的组件，返回符合条件的组件列表

默认接口请求频率限制：10000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：SearchKBComponent。
Version	是	String	公共参数，本接口取值：2021-08-11。
Region	否	String	公共参数，此参数为可选参数。
Query	是	String	需要搜索的组件名 示例值：openssl
Protocol	否	String	需要搜索的组件类型 示例值：generic
PageNumber	否	Integer	分页参数，从 0 开始 示例值：0
PageSize	否	Integer	分页参数，设置每页返回的结果数量 示例值：20

3. 输出参数

参数名称	类型	描述
ComponentList	Array of Component	满足搜索条件的组件列表
Total	Integer	满足搜索条件的总个数 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 搜索知识库组件列表

搜索ristretto相关组件

输入示例

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

<公共请求参数>

```
{
"Query": "ristretto"
}
```

输出示例


```
{
"Response": {
"ComponentList": [
{
"CodeLocationList": null,
"Homepage": "",
"LicenseExpression": "bsd-new",
"NicknameList": null,
"PURL": {
"Name": "ristretto",
"Namespace": "@polymer",
"Protocol": "npm",
"Qualifiers": [],
"Subpath": "",
"Version": ""
},
"Summary": "An extensible test runner 🍵"
},
{
"CodeLocationList": [
"github.com/bobotu/ristretto"
],
"Homepage": "",
"LicenseExpression": "apache-2.0",
"NicknameList": null,
"PURL": {
"Name": "ristretto",
"Namespace": "github.com/bobotu",
"Protocol": "golang",
"Qualifiers": [],
"Subpath": "",
"Version": ""
},
"Summary": ""
},
{
"CodeLocationList": [
"github.com/RangelReale/trcache"
],
"Homepage": "",
"LicenseExpression": "mit",
"NicknameList": null,
"PURL": {
"Name": "ristretto",
"Namespace": "github.com/rangelreale/trcache/cache",
"Protocol": "golang",
"Qualifiers": [],
```

```
"Subpath": "",
"Version": ""
},
"Summary": "This section is empty"
},
{
"CodeLocationList": [
"github.com/dgraph-io/ristretto"
],
"Homepage": "",
"LicenseExpression": "apache-2.0",
"NicknameList": null,
"PURL": {
"Name": "ristretto",
"Namespace": "github.com/dgraph-io",
"Protocol": "golang",
"Qualifiers": [],
"Subpath": "",
"Version": ""
},
"Summary": "Ristretto is a fast, concurrent cache library built with a focus on performance and correctness"
},
{
"CodeLocationList": null,
"Homepage": "https://github.com/Benli11/ristretto",
"LicenseExpression": "",
"NicknameList": null,
"PURL": {
"Name": "ristretto",
"Namespace": "",
"Protocol": "pypi",
"Qualifiers": [],
"Subpath": "",
"Version": ""
},
"Summary": "ristretto: Randomized Dimension Reduction Library"
},
{
"CodeLocationList": [
"github.com/etecs-ru/ristretto"
],
"Homepage": "",
"LicenseExpression": "apache-2.0",
"NicknameList": null,
"PURL": {
"Name": "ristretto",
"Namespace": "github.com/etecs-ru",
"Protocol": "golang",
"Qualifiers": [],
"Subpath": "",
"Version": ""
},
"Summary": "This fork applies the selected PRs from the original repo"
},
{
```

```
"CodeLocationList": [
"github.com/knocknote/gocache"
],
"Homepage": "",
"LicenseExpression": "mit",
"NicknameList": null,
"PURL": {
"Name": "ristretto",
"Namespace": "github.com/knocknote/gocache/store",
"Protocol": "golang",
"Qualifiers": [],
"Subpath": "",
"Version": ""
},
"Summary": ""
},
{
"CodeLocationList": [
"github.com/outcaste-io/ristretto"
],
"Homepage": "",
"LicenseExpression": "apache-2.0",
"NicknameList": null,
"PURL": {
"Name": "ristretto",
"Namespace": "github.com/outcaste-io",
"Protocol": "golang",
"Qualifiers": [],
"Subpath": "",
"Version": ""
},
"Summary": "This is a fork of dgraph-io/ristretto, maintained by @manishrjain"
},
{
"CodeLocationList": [
"github.com/sergiub32/ristretto"
],
"Homepage": "",
"LicenseExpression": "apache-2.0",
"NicknameList": null,
"PURL": {
"Name": "ristretto",
"Namespace": "github.com/sergiub32",
"Protocol": "golang",
"Qualifiers": [],
"Subpath": "",
"Version": ""
},
"Summary": ""
},
{
"CodeLocationList": [
"http://git.xfce.org/apps/ristretto"
],
"Homepage": "",
```

```
"LicenseExpression": "gpl-2.0",
"NicknameList": null,
"PURL": {
"Name": "ristretto",
"Namespace": "apps",
"Protocol": "generic",
"Qualifiers": [
{
"Key": "vcs_url",
"Value": "http://git.xfce.org/apps/ristretto"
}
],
"Subpath": "",
"Version": ""
},
"Summary": " Ristretto is a fast and lightweight picture-viewer for the Xfce desktop\n environment.\nOriginal-Maintaine
r: Debian Xfce Maintainers <pkg-xfce-devel@lists.alioth.debian.org>\n"
},
{
"CodeLocationList": [
"github.com/revcontent-production/ristretto"
],
"Homepage": "",
"LicenseExpression": "apache-2.0",
"NicknameList": null,
"PURL": {
"Name": "ristretto",
"Namespace": "github.com/revcontent-production",
"Protocol": "golang",
"Qualifiers": [],
"Subpath": "",
"Version": ""
},
"Summary": "Ristretto is a fast, concurrent cache library built with a focus on performance and correctness"
},
{
"CodeLocationList": [
"github.com/ben-han-cn/ristretto"
],
"Homepage": "",
"LicenseExpression": "apache-2.0",
"NicknameList": null,
"PURL": {
"Name": "ristretto",
"Namespace": "github.com/ben-han-cn",
"Protocol": "golang",
"Qualifiers": [],
"Subpath": "",
"Version": ""
},
"Summary": ""
},
{
"CodeLocationList": [
"github.com/gofiber/storage"
```

```
],
"Homepage": "",
"LicenseExpression": "mit",
"NicknameList": null,
"PURL": {
"Name": "ristretto",
"Namespace": "github.com/gofiber/storage",
"Protocol": "golang",
"Qualifiers": [],
"Subpath": "",
"Version": ""
},
"Summary": "ufeff# Ristretto\nA Memory-bound storage driver using dgraph-io/ristretto"
},
{
"CodeLocationList": [
"github.com/flarebuild/ristretto"
],
"Homepage": "",
"LicenseExpression": "apache-2.0",
"NicknameList": null,
"PURL": {
"Name": "ristretto",
"Namespace": "github.com/flarebuild",
"Protocol": "golang",
"Qualifiers": [],
"Subpath": "",
"Version": ""
},
"Summary": ""
},
{
"CodeLocationList": [
"github.com/impasse/ristretto"
],
"Homepage": "",
"LicenseExpression": "apache-2.0",
"NicknameList": null,
"PURL": {
"Name": "ristretto",
"Namespace": "github.com/impasse",
"Protocol": "golang",
"Qualifiers": [],
"Subpath": "",
"Version": ""
},
"Summary": ""
},
{
"CodeLocationList": [
"github.com/coicoichip/ristretto"
],
"Homepage": "",
"LicenseExpression": "apache-2.0",
"NicknameList": null,
```

```
"PURL": {
"Name": "ristretto",
"Namespace": "github.com/coicoichip",
"Protocol": "golang",
"Qualifiers": [],
"Subpath": "",
"Version": ""
},
"Summary": ""
},
{
"CodeLocationList": [
"github.com/paivagustavo/ristretto"
],
"Homepage": "",
"LicenseExpression": "",
"NicknameList": null,
"PURL": {
"Name": "ristretto",
"Namespace": "github.com/paivagustavo",
"Protocol": "golang",
"Qualifiers": [],
"Subpath": "",
"Version": ""
},
"Summary": "Ristretto is a fast, concurrent cache library built with a focus on performance and correctness"
},
{
"CodeLocationList": [
"github.com/alexsergivan/blog-examples"
],
"Homepage": "",
"LicenseExpression": "",
"NicknameList": null,
"PURL": {
"Name": "ristretto",
"Namespace": "github.com/alexsergivan/blog-examples",
"Protocol": "golang",
"Qualifiers": [],
"Subpath": "",
"Version": ""
},
"Summary": ""
},
{
"CodeLocationList": [
"github.com/ahiho/xcache"
],
"Homepage": "",
"LicenseExpression": "",
"NicknameList": null,
"PURL": {
"Name": "ristretto",
"Namespace": "github.com/ahiho/xcache/driver",
"Protocol": "golang",
```

```
"Qualifiers": [],
"Subpath": "",
"Version": ""
},
"Summary": ""
},
{
"CodeLocationList": [
"github.com/aryehlev/ristretto"
],
"Homepage": "",
"LicenseExpression": "apache-2.0",
"NicknameList": null,
"PURL": {
"Name": "ristretto",
"Namespace": "github.com/aryehlev",
"Protocol": "golang",
"Qualifiers": [],
"Subpath": "",
"Version": ""
},
"Summary": "Ristretto is a fast, concurrent cache library built with a focus on performance and correctness"
}
],
"RequestId": "3a979aa9-2612-4155-8b1b-cf850e71c0f3",
"Total": 41
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.AccountNotEnough	账户流量余额不足。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
MissingParameter	缺少参数错误。

查询知识库组件信息

最近更新时间：2024-10-29 21:02:27

1. 接口描述

接口请求域名： bsca.tencentcloudapi.com 。

本接口(DescribeKBComponent)用于在知识库中查询开源组件信息。本接口根据用户输入的PURL在知识库中寻找对应的开源组件，其中Name为必填字段。

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：DescribeKBComponent。
Version	是	String	公共参数，本接口取值：2021-08-11。
Region	否	String	公共参数，本接口不需要传递此参数。
PURL	是	PURL	组件的PURL

3. 输出参数

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

4. 示例

示例1 查询openssl组件信息

输入示例

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

{
"PURL": {
"Name": "openssl"
}
}
```

输出示例

```
{
"Response": {
"Component": {
"CodeLocationList": [
"https://github.com/openssl/openssl.git",
"https://android.googlesource.com/platform/external/openssl"
],
"Homepage": "https://www.openssl.org",
"LastUpdateTime": "2023-09-11 22:08:48.000000",
"LicenseExpression": "openssl-ssleay",
"NicknameList": null,
"PURL": {
"Name": "openssl",
"Namespace": "",
"Protocol": "generic",
"Qualifiers": [],
"Subpath": "",
"Version": "1.0.0"
},
"Summary": "OpenSSL is a robust, commercial-grade, full-featured Open Source Toolkit for the Transport Layer Security (TLS) protocol formerly known as the Secure Sockets Layer (SSL) protocol. The protocol implementation is based on a full-strength general purpose cryptographic library, which can also be used stand-alone.",
"TagList": [
"network"
],
"VersionInfo": {
"CopyrightList": [
"Copyright (c) 1998-2008 The OpenSSL Project",
"Copyright (c) 1995-1998 Eric Young (eay@cryptsoft.com)",
"holder is Tim Hudson (tjh@cryptsoft.com)"
],
"PublishTime": "2015-01-22 09:46:52.000000",
"TagList": [
"copyright_updated"
]
}
},
"RequestId": "deadc0de-3ab7-45dd-9def-6b3e087dd354"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.AccountNotEnough	账户流量余额不足。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
ResourceNotFound	资源不存在。

查询知识库组件漏洞

最近更新时间：2024-10-29 21:02:27

1. 接口描述

接口请求域名： bsca.tencentcloudapi.com 。

本接口(DescribeKBComponentVulnerability)用于在知识库中查询开源组件的漏洞信息。

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：DescribeKBComponentVulnerability。
Version	是	String	公共参数，本接口取值：2021-08-11。
Region	否	String	公共参数，本接口不需要传递此参数。
PURL	是	PURL	组件的PURL，其中Name和Version为必填字段
Language	否	String	语言，ZH或EN 示例值：ZH

3. 输出参数

参数名称	类型	描述
VulnerabilityList	Array of ComponentVulnerabilityUnion	漏洞信息列表 注意：此字段可能返回 null，表示取不到有效值。
PURL	PURL	组件purl
RecommendedVersion	String	推荐版本，当前版本中的所有漏洞都修复了的版本 示例值：1.1.12
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openssl 1.1.1的漏洞信息

输入示例

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

{
"PURL": {
```

```
"Name": "xxl-job-core",
"Protocol": "maven",
"Namespace": "com.xuxueli",
"Version": "2.2.0"
}
}
```


输出示例


```
{
"Response": {
"PURL": {
"Name": "com.xuxueli:xxl-job-core",
"Namespace": "",
"Protocol": "maven",
"Qualifiers": [],
"Subpath": "",
"Version": "2.2.0"
},
"RecommendedVersion": "2.4.1",
"VulnerabilityList": [
{
"Summary": {
"ArchitectureList": [],
"CNNVDID": "CNNVD-202408-1426",
"CNVDID": "",
"CVEID": "CVE-2024-42681",
"IsSuggest": true,
"Name": "XXL-JOB 安全漏洞",
"PatchUrlList": null,
"Severity": "High",
"VulID": "pcmgr-513695"
},
"SummaryInComponent": {
"AffectedComponent": "com.xuxueli:xxl-job-core",
"AffectedVersion": "2.2.0",
"CanBeFixed": true,
"FixedVersion": ">2.4.1",
"PURL": {
"Name": "com.xuxueli:xxl-job-core",
"Namespace": "",
"Protocol": "maven",
"Qualifiers": [],
"Subpath": "",
"Version": "2.2.0"
},
"RiskLevel": "High"
}
},
{
"Summary": {
"ArchitectureList": [],
"CNNVDID": "CNNVD-202012-1604",
"CNVDID": "CNVD-2021-44699",
"CVEID": "CVE-2020-29204",
```

```
"IsSuggest": false,
"Name": "XXL-JOB跨站脚本漏洞（CVE-2020-29204）",
"PatchUrlList": null,
"Severity": "Medium",
"VulID": "pcmgr-304614"
},
"SummaryInComponent": {
"AffectedComponent": "com.xuxueli:xxl-job-core",
"AffectedVersion": "2.2.0",
"CanBeFixed": true,
"FixedVersion": ">2.2.0,2.3.0",
"PURL": {
"Name": "com.xuxueli:xxl-job-core",
"Namespace": "",
"Protocol": "maven",
"Qualifiers": [],
"Subpath": "",
"Version": "2.2.0"
},
"RiskLevel": "Medium"
}
},
{
"Summary": {
"ArchitectureList": [],
"CNNVDID": "CNNVD-202209-2891",
"CNVDID": "",
"CVEID": "CVE-2022-40929",
"IsSuggest": true,
"Name": "XXL-JOB 命令注入漏洞",
"PatchUrlList": null,
"Severity": "Critical",
"VulID": "pcmgr-368382"
},
"SummaryInComponent": {
"AffectedComponent": "com.xuxueli:xxl-job-core",
"AffectedVersion": "2.2.0",
"CanBeFixed": true,
"FixedVersion": ">2.2.0",
"PURL": {
"Name": "com.xuxueli:xxl-job-core",
"Namespace": "",
"Protocol": "maven",
"Qualifiers": [],
"Subpath": "",
"Version": "2.2.0"
},
"RiskLevel": "Critical"
}
},
{
"Summary": {
"ArchitectureList": [],
"CNNVDID": "",
"CNVDID": "CNVD-2022-81229",
```

```
"CVEID": "CVE-2022-43183",
"IsSuggest": true,
"Name": "XXL-JOB代码问题漏洞(CVE-2022-43183)",
"PatchUrlList": null,
"Severity": "High",
"VulID": "pcmgr-378820"
},
"SummaryInComponent": {
"AffectedComponent": "com.xuxueli:xxl-job-core",
"AffectedVersion": "2.2.0",
"CanBeFixed": true,
"FixedVersion": ">2.3.0,2.3.1",
"PURL": {
"Name": "com.xuxueli:xxl-job-core",
"Namespace": "",
"Protocol": "maven",
"Qualifiers": [],
"Subpath": "",
"Version": "2.2.0"
},
"RiskLevel": "High"
}
},
{
"Summary": {
"ArchitectureList": [],
"CNNVDID": "CNNVD-202404-775",
"CNVDID": "",
"CVEID": "CVE-2024-3366",
"IsSuggest": false,
"Name": "XXL-JOB 注入漏洞(CVE-2024-3366)",
"PatchUrlList": null,
"Severity": "High",
"VulID": "pcmgr-485156"
},
"SummaryInComponent": {
"AffectedComponent": "com.xuxueli:xxl-job-core",
"AffectedVersion": "2.2.0",
"CanBeFixed": true,
"FixedVersion": ">2.4.0,2.4.1",
"PURL": {
"Name": "com.xuxueli:xxl-job-core",
"Namespace": "",
"Protocol": "maven",
"Qualifiers": [],
"Subpath": "",
"Version": "2.2.0"
},
"RiskLevel": "High"
}
}
],
"RequestId": "3eed4bda-8f2c-444f-bac8-d64555eb0e05"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.AccountNotEnough	账户流量余额不足。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
MissingParameter	缺少参数错误。

查询知识库许可证信息

最近更新时间：2024-03-12 01:11:52

1. 接口描述

接口请求域名： bsca.tencentcloudapi.com 。

本接口(DescribeKBLicense)用于在知识库中查询许可证信息。

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：DescribeKBLicense。
Version	是	String	公共参数，本接口取值：2021-08-11。
Region	否	String	公共参数，本接口不需要传递此参数。
LicenseExpression	否	String	License表达式 示例值：apache-2.0

3. 输出参数

参数名称	类型	描述
LicenseList	Array of LicenseUnion	许可证列表 注意：此字段可能返回 null，表示取不到有效值。
NormalizedLicenseExpression	String	用于匹配的License表达式 示例值：apache-2.0
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 apache-2.0 许可证信息

输入示例

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

{
"LicenseExpression": "apache-2.0"
}
```

输出示例

```
{
"Response": {
"LicenseList": [
{
"LicenseDetail": {
"ConditionSet": [
{
"Description": "A copy of the license and copyright notice must be included with the licensed material.",
"Name": "License and copyright notice"
},
{
"Description": "Changes made to the licensed material must be documented.",
"Name": "State changes"
}
],
"Content": " Apache License\n Version 2.0, January 2004\n",
"ForbiddenSet": [
{
"Description": "This license explicitly states that it does NOT grant trademark rights, even though licenses without su
ch a statement probably do not grant any implicit trademark rights.",
"Name": "Trademark use"
},
{
"Description": "This license includes a limitation of liability.",
"Name": "Liability"
},
{
"Description": "This license explicitly states that it does NOT provide any warranty.",
"Name": "Warranty"
}
],
"PermissionSet": [
{
"Description": "The licensed material and derivatives may be used for commercial purposes.",
"Name": "Commercial use"
},
{
"Description": "The licensed material may be modified.",
"Name": "Modification"
},
{
"Description": "The licensed material may be distributed.",
"Name": "Distribution"
},
{
"Description": "This license explicitly states that it does NOT grant any rights in the patents of contributors.",
"Name": "Patent use"
},
{
"Description": "The licensed material may be used and modified in private.",
"Name": "Private use"
}
]
},
"LicenseSummary": {
```

```
"Key": "apache-2.0",
"Name": "Apache License 2.0",
"Risk": "MediumRisk",
"SPDXKey": "Apache-2.0",
"ShortName": "Apache 2.0",
"Source": "https://spdx.org/licenses/Apache-2.0.html"
}
}
],
"NormalizedLicenseExpression": "apache-2.0",
"RequestId": "26112148-876a-4bc7-a427-1985371c65fd"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.AccountNotEnough	账户流量余额不足。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
MissingParameter	缺少参数错误。
ResourceNotFound	资源不存在。

查询知识库漏洞详情列表

最近更新时间：2024-04-16 01:05:25

1. 接口描述

接口请求域名： bsca.tencentcloudapi.com 。

本接口(DescribeKBVulnerability)用于在知识库中查询漏洞详细信息，支持根据CVE、Vul ID、CNVD ID、CNNVD ID查询。

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：DescribeKBVulnerability。
Version	是	String	公共参数，本接口取值：2021-08-11。
Region	否	String	公共参数，本接口不需要传递此参数。
CVEID.N	否	Array of String	根据CVE ID查询（不能与其他参数同时存在） 示例值：["CVE-2020-1971", "CVE-2019-1551"]
VulID.N	否	Array of String	根据Vul ID查询（不能与其他参数同时存在） 示例值：["pcmgr-303514"]
CNVDID.N	否	Array of String	根据CNVD ID查询（不能与其他参数同时存在） 示例值：["CNVD-2021-04811"]
CNNVDID.N	否	Array of String	根据CNNVD ID查询（不能与其他参数同时存在） 示例值：["CNNVD-202108-1945"]
Language	否	String	语言，ZH或EN 示例值：ZH

3. 输出参数

参数名称	类型	描述
VulnerabilityDetailList	Array of VulnerabilityUnion	漏洞详细信息列表 注意：此字段可能返回 null，表示取不到有效值。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询CVE-2019-1551的漏洞详细信息

查询CVE-2019-1551的漏洞详细信息

输入示例

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

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

{
"CVEID": [
"CVE-2019-1551"
]
}
```

输出示例


```
{
"Response": {
"RequestId": "8db27592-1fa7-4bb2-bbc0-7a962635a0eb",
"VulnerabilityDetailList": [
{
"Detail": {
"AffectedComponentList": [
{
"AffectedVersionList": [
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
],
"Name": "openssl-libs"
},
{
"AffectedVersionList": [
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
],
"Name": "openssl-devel"
},
{
"AffectedVersionList": [
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
],
"Name": "openssl098e"
},
{
"AffectedVersionList": [
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
```

```
],
"Name": "libssl1.1"
},
{
"AffectedVersionList": [
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
],
"Name": "libcrypto1.1"
},
{
"AffectedVersionList": [
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
],
"Name": "ssl_client"
},
{
"AffectedVersionList": [
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
],
"Name": "openssl1.0"
},
{
"AffectedVersionList": [
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
],
"Name": "libssl1.0.2"
},
{
"AffectedVersionList": [
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
],
"Name": "libssl-dev"
},
{
"AffectedVersionList": [
```

```
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
],
"Name": "libssl1.0.0"
},
{
"AffectedVersionList": [
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
],
"Name": "openssl098"
},
{
"AffectedVersionList": [
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
],
"Name": "openssl-perl"
},
{
"AffectedVersionList": [
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
],
"Name": "openssl-universal"
},
{
"AffectedVersionList": [
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
],
"Name": "openssl-osx"
},
{
"AffectedVersionList": [
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
```

```
],
"Name": "openssl-static"
},
{
"AffectedVersionList": [
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
],
"Name": "mingw32-openssl"
},
{
"AffectedVersionList": [
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
],
"Name": "compat-openssl10"
},
{
"AffectedVersionList": [
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
],
"Name": "mingw64-openssl"
},
{
"AffectedVersionList": [
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
],
"Name": "openssl-debugsource"
},
{
"AffectedVersionList": [
"1.1.1<=version<=1.1.1d",
"1.0.2<=version<=1.0.2t"
],
"FixedVersionList": [
">1.0.2t,>1.1.1d"
],
"Name": "openssl"
},
{
"AffectedVersionList": [
```

```
"version<0:1.0.2u-1.ph1"
],
"FixedVersionList": [
"version>=0:1.0.2u-1.ph1"
],
"Name": "(photon:1.0) openssl"
},
{
"AffectedVersionList": [
"version<0:1.0.2u-1.ph1"
],
"FixedVersionList": [
"version>=0:1.0.2u-1.ph1"
],
"Name": "(photon:1.0) openssl-c_rehash"
},
{
"AffectedVersionList": [
"version<0:1.0.2u-1.ph1"
],
"FixedVersionList": [
"version>=0:1.0.2u-1.ph1"
],
"Name": "(photon:1.0) openssl-devel"
},
{
"AffectedVersionList": [
"version<0:1.0.2u-1.ph1"
],
"FixedVersionList": [
"version>=0:1.0.2u-1.ph1"
],
"Name": "(photon:1.0) openssl-perl"
},
{
"AffectedVersionList": [
"version<1.1.1d-0+deb10u5"
],
"FixedVersionList": [
"version>=1.1.1d-0+deb10u5"
],
"Name": "(debian:10) openssl"
},
{
"AffectedVersionList": [
"version<1.1.1e-1"
],
"FixedVersionList": [
"version>=1.1.1e-1"
],
"Name": "(debian:11) openssl"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-3.14"
```

```
],
"FixedVersionList": [
"version>=0:1.0.2p-3.14"
],
"Name": "(sles:12) libopenssl-1_0_0-devel"
},
{
"AffectedVersionList": [
"version<0:1.0.2j-60.60"
],
"FixedVersionList": [
"version>=0:1.0.2j-60.60"
],
"Name": "(sles:12) libopenssl-devel"
},
{
"AffectedVersionList": [
"version<0:1.1.1d-2.20"
],
"FixedVersionList": [
"version>=0:1.1.1d-2.20"
],
"Name": "(sles:12) libopenssl1_1"
},
{
"AffectedVersionList": [
"version<0:1.1.1d-2.20"
],
"FixedVersionList": [
"version>=0:1.1.1d-2.20"
],
"Name": "(sles:12) libopenssl1_1-32bit"
},
{
"AffectedVersionList": [
"version<0:1.0.2j-60.60"
],
"FixedVersionList": [
"version>=0:1.0.2j-60.60"
],
"Name": "(sles:12) openssl"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-3.14"
],
"FixedVersionList": [
"version>=0:1.0.2p-3.14"
],
"Name": "(sles:12) openssl-1_0_0"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-3.14"
],
```

```
"FixedVersionList": [
"version>=0:1.0.2p-3.14"
],
"Name": "(sles:12) openssl-1_0_0-doc"
},
{
"AffectedVersionList": [
"version<0:1.1.1d-2.20"
],
"FixedVersionList": [
"version>=0:1.1.1d-2.20"
],
"Name": "(sles:12) openssl-1_1"
},
{
"AffectedVersionList": [
"version<0:1.0.2j-60.60"
],
"FixedVersionList": [
"version>=0:1.0.2j-60.60"
],
"Name": "(sles:12) openssl-doc"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-lp151.5.13"
],
"FixedVersionList": [
"version>=0:1.0.2p-lp151.5.13"
],
"Name": "(opensuse-leap:15.1) libopenssl-1_0_0-devel"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-lp151.5.13"
],
"FixedVersionList": [
"version>=0:1.0.2p-lp151.5.13"
],
"Name": "(opensuse-leap:15.1) libopenssl-1_0_0-devel-32bit"
},
{
"AffectedVersionList": [
"version<0:1.1.0i-lp151.8.6"
],
"FixedVersionList": [
"version>=0:1.1.0i-lp151.8.6"
],
"Name": "(opensuse-leap:15.1) libopenssl-1_1-devel"
},
{
"AffectedVersionList": [
"version<0:1.1.0i-lp151.8.6"
],
"FixedVersionList": [
```

```
"version>=0:1.1.0i-lp151.8.6"
],
"Name": "(opensuse-leap:15.1) libopenssl-1_1-devel-32bit"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-lp151.5.13"
],
"FixedVersionList": [
"version>=0:1.0.2p-lp151.5.13"
],
"Name": "(opensuse-leap:15.1) libopenssl1_0_0"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-lp151.5.13"
],
"FixedVersionList": [
"version>=0:1.0.2p-lp151.5.13"
],
"Name": "(opensuse-leap:15.1) libopenssl1_0_0-32bit"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-lp151.5.13"
],
"FixedVersionList": [
"version>=0:1.0.2p-lp151.5.13"
],
"Name": "(opensuse-leap:15.1) libopenssl1_0_0-hmac"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-lp151.5.13"
],
"FixedVersionList": [
"version>=0:1.0.2p-lp151.5.13"
],
"Name": "(opensuse-leap:15.1) libopenssl1_0_0-hmac-32bit"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-lp151.5.13"
],
"FixedVersionList": [
"version>=0:1.0.2p-lp151.5.13"
],
"Name": "(opensuse-leap:15.1) libopenssl1_0_0-steam"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-lp151.5.13"
],
"FixedVersionList": [
"version>=0:1.0.2p-lp151.5.13"
```

```
],
"Name": "(opensuse-leap:15.1) libopenssl1_0_0-steam-32bit"
},
{
"AffectedVersionList": [
"version<0:1.1.0i-lp151.8.6"
],
"FixedVersionList": [
"version>=0:1.1.0i-lp151.8.6"
],
"Name": "(opensuse-leap:15.1) libopenssl1_1"
},
{
"AffectedVersionList": [
"version<0:1.1.0i-lp151.8.6"
],
"FixedVersionList": [
"version>=0:1.1.0i-lp151.8.6"
],
"Name": "(opensuse-leap:15.1) libopenssl1_1-32bit"
},
{
"AffectedVersionList": [
"version<0:1.1.0i-lp151.8.6"
],
"FixedVersionList": [
"version>=0:1.1.0i-lp151.8.6"
],
"Name": "(opensuse-leap:15.1) libopenssl1_1-hmac"
},
{
"AffectedVersionList": [
"version<0:1.1.0i-lp151.8.6"
],
"FixedVersionList": [
"version>=0:1.1.0i-lp151.8.6"
],
"Name": "(opensuse-leap:15.1) libopenssl1_1-hmac-32bit"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-lp151.5.13"
],
"FixedVersionList": [
"version>=0:1.0.2p-lp151.5.13"
],
"Name": "(opensuse-leap:15.1) openssl-1_0_0"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-lp151.5.13"
],
"FixedVersionList": [
"version>=0:1.0.2p-lp151.5.13"
],
```

```
"Name": "(opensuse-leap:15.1) openssl-1_0_0-cavs"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-lp151.5.13"
],
"FixedVersionList": [
"version>=0:1.0.2p-lp151.5.13"
],
"Name": "(opensuse-leap:15.1) openssl-1_0_0-doc"
},
{
"AffectedVersionList": [
"version<0:1.1.0i-lp151.8.6"
],
"FixedVersionList": [
"version>=0:1.1.0i-lp151.8.6"
],
"Name": "(opensuse-leap:15.1) openssl-1_1"
},
{
"AffectedVersionList": [
"version<0:1.1.0i-lp151.8.6"
],
"FixedVersionList": [
"version>=0:1.1.0i-lp151.8.6"
],
"Name": "(opensuse-leap:15.1) openssl-1_1-doc"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-3.25"
],
"FixedVersionList": [
"version>=0:1.0.2p-3.25"
],
"Name": "(sles:15) libopenssl-1_0_0-devel"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-3.25"
],
"FixedVersionList": [
"version>=0:1.0.2p-3.25"
],
"Name": "(sles:15) libopenssl-1_0_0-devel-32bit"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-3.25"
],
"FixedVersionList": [
"version>=0:1.0.2p-3.25"
],
"Name": "(sles:15) libopenssl1_0_0"
```

```
},
{
"AffectedVersionList": [
"version<0:1.0.2p-3.25"
],
"FixedVersionList": [
"version>=0:1.0.2p-3.25"
],
"Name": "(sles:15) libopenssl1_0_0-32bit"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-3.25"
],
"FixedVersionList": [
"version>=0:1.0.2p-3.25"
],
"Name": "(sles:15) libopenssl1_0_0-hmac"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-3.25"
],
"FixedVersionList": [
"version>=0:1.0.2p-3.25"
],
"Name": "(sles:15) libopenssl1_0_0-hmac-32bit"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-3.25"
],
"FixedVersionList": [
"version>=0:1.0.2p-3.25"
],
"Name": "(sles:15) libopenssl1_0_0-steam"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-3.25"
],
"FixedVersionList": [
"version>=0:1.0.2p-3.25"
],
"Name": "(sles:15) libopenssl1_0_0-steam-32bit"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-3.25"
],
"FixedVersionList": [
"version>=0:1.0.2p-3.25"
],
"Name": "(sles:15) openssl-1_0_0"
},
```

```
{
"AffectedVersionList": [
"version<0:1.0.2p-3.25"
],
"FixedVersionList": [
"version>=0:1.0.2p-3.25"
],
"Name": "(sles:15) openssl-1_0_0-cavs"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-3.25"
],
"FixedVersionList": [
"version>=0:1.0.2p-3.25"
],
"Name": "(sles:15) openssl-1_0_0-doc"
},
{
"AffectedVersionList": [
"version<0:1.0.2g-1ubuntu4.16"
],
"FixedVersionList": [
"version>=0:1.0.2g-1ubuntu4.16"
],
"Name": "(ubuntu:16.04) libssl1.0.0"
},
{
"AffectedVersionList": [
"version<0:1.0.2g-1ubuntu4.16"
],
"FixedVersionList": [
"version>=0:1.0.2g-1ubuntu4.16"
],
"Name": "(ubuntu:16.04) openssl"
},
{
"AffectedVersionList": [
"version<0:1.0.2n-1ubuntu5.4"
],
"FixedVersionList": [
"version>=0:1.0.2n-1ubuntu5.4"
],
"Name": "(ubuntu:18.04) libssl1.0.0"
},
{
"AffectedVersionList": [
"version<0:1.1.1-1ubuntu2.1~18.04.6"
],
"FixedVersionList": [
"version>=0:1.1.1-1ubuntu2.1~18.04.6"
],
"Name": "(ubuntu:18.04) libssl1.1"
},
{
```

```
"AffectedVersionList": [
"version<0:1.1.1-1ubuntu2.1~18.04.6"
],
"FixedVersionList": [
"version>=0:1.1.1-1ubuntu2.1~18.04.6"
],
"Name": "(ubuntu:18.04) openssl"
},
{
"AffectedVersionList": [
"version<0:1.0.2n-1ubuntu5.4"
],
"FixedVersionList": [
"version>=0:1.0.2n-1ubuntu5.4"
],
"Name": "(ubuntu:18.04) openssl1.0"
},
{
"AffectedVersionList": [
"version<1.1.1c-1ubuntu4.1"
],
"FixedVersionList": [
"version>=1.1.1c-1ubuntu4.1"
],
"Name": "(ubuntu:19.10) openssl"
},
{
"AffectedVersionList": [
"version<0:1.1.1d-2.ph2"
],
"FixedVersionList": [
"version>=0:1.1.1d-2.ph2"
],
"Name": "(photon:2.0) nxtgn-openssl"
},
{
"AffectedVersionList": [
"version<0:1.1.1d-2.ph2"
],
"FixedVersionList": [
"version>=0:1.1.1d-2.ph2"
],
"Name": "(photon:2.0) nxtgn-openssl-c_rehash"
},
{
"AffectedVersionList": [
"version<0:1.1.1d-2.ph2"
],
"FixedVersionList": [
"version>=0:1.1.1d-2.ph2"
],
"Name": "(photon:2.0) nxtgn-openssl-devel"
},
{
"AffectedVersionList": [
```

```
"version<0:1.1.1d-2.ph2"
],
"FixedVersionList": [
"version>=0:1.1.1d-2.ph2"
],
"Name": "(photon:2.0) nxtgn-openssl-perl"
},
{
"AffectedVersionList": [
"version<0:1.0.2u-2.ph2"
],
"FixedVersionList": [
"version>=0:1.0.2u-2.ph2"
],
"Name": "(photon:2.0) openssl"
},
{
"AffectedVersionList": [
"version<0:1.0.2u-2.ph2"
],
"FixedVersionList": [
"version>=0:1.0.2u-2.ph2"
],
"Name": "(photon:2.0) openssl-c_rehash"
},
{
"AffectedVersionList": [
"version<0:1.0.2u-2.ph2"
],
"FixedVersionList": [
"version>=0:1.0.2u-2.ph2"
],
"Name": "(photon:2.0) openssl-devel"
},
{
"AffectedVersionList": [
"version<0:1.0.2u-2.ph2"
],
"FixedVersionList": [
"version>=0:1.0.2u-2.ph2"
],
"Name": "(photon:2.0) openssl-perl"
},
{
"AffectedVersionList": [
"version<0:1.1.1f-1ubuntu1"
],
"FixedVersionList": [
"version>=0:1.1.1f-1ubuntu1"
],
"Name": "(ubuntu:20.04) libssl1.1"
},
{
"AffectedVersionList": [
"version<0:1.1.1f-1ubuntu1"
```

```
],
"FixedVersionList": [
"version>=0:1.1.1f-1ubuntu1"
],
"Name": "(ubuntu:20.04) openssl"
},
{
"AffectedVersionList": [
"version<1.0.2k-19.amzn2.0.7"
],
"FixedVersionList": [
"version>=1.0.2k-19.amzn2.0.7"
],
"Name": "(amzn:2) openssl"
},
{
"AffectedVersionList": [
"version<1.0.2k-19.amzn2.0.7"
],
"FixedVersionList": [
"version>=1.0.2k-19.amzn2.0.7"
],
"Name": "(amzn:2) openssl-debuginfo"
},
{
"AffectedVersionList": [
"version<1.0.2k-19.amzn2.0.7"
],
"FixedVersionList": [
"version>=1.0.2k-19.amzn2.0.7"
],
"Name": "(amzn:2) openssl-devel"
},
{
"AffectedVersionList": [
"version<1.0.2k-19.amzn2.0.7"
],
"FixedVersionList": [
"version>=1.0.2k-19.amzn2.0.7"
],
"Name": "(amzn:2) openssl-libs"
},
{
"AffectedVersionList": [
"version<1.0.2k-19.amzn2.0.7"
],
"FixedVersionList": [
"version>=1.0.2k-19.amzn2.0.7"
],
"Name": "(amzn:2) openssl-perl"
},
{
"AffectedVersionList": [
"version<1.0.2k-19.amzn2.0.7"
],
```

```
"FixedVersionList": [
"version>=1.0.2k-19.amzn2.0.7"
],
"Name": "(amzn:2) openssl-static"
},
{
"AffectedVersionList": [
"version<0:1.1.1d-2.ph3"
],
"FixedVersionList": [
"version>=0:1.1.1d-2.ph3"
],
"Name": "(photon:3.0) nxtgn-openssl"
},
{
"AffectedVersionList": [
"version<0:1.1.1d-2.ph3"
],
"FixedVersionList": [
"version>=0:1.1.1d-2.ph3"
],
"Name": "(photon:3.0) nxtgn-openssl-c_rehash"
},
{
"AffectedVersionList": [
"version<0:1.1.1d-2.ph3"
],
"FixedVersionList": [
"version>=0:1.1.1d-2.ph3"
],
"Name": "(photon:3.0) nxtgn-openssl-devel"
},
{
"AffectedVersionList": [
"version<0:1.1.1d-2.ph3"
],
"FixedVersionList": [
"version>=0:1.1.1d-2.ph3"
],
"Name": "(photon:3.0) nxtgn-openssl-perl"
},
{
"AffectedVersionList": [
"version<0:1.0.2u-1.ph3"
],
"FixedVersionList": [
"version>=0:1.0.2u-1.ph3"
],
"Name": "(photon:3.0) openssl"
},
{
"AffectedVersionList": [
"version<0:1.0.2u-1.ph3"
],
"FixedVersionList": [
```

```
"version>=0:1.0.2u-1.ph3"
],
"Name": "(photon:3.0) openssl-c_rehash"
},
{
"AffectedVersionList": [
"version<0:1.0.2u-1.ph3"
],
"FixedVersionList": [
"version>=0:1.0.2u-1.ph3"
],
"Name": "(photon:3.0) openssl-devel"
},
{
"AffectedVersionList": [
"version<0:1.0.2u-1.ph3"
],
"FixedVersionList": [
"version>=0:1.0.2u-1.ph3"
],
"Name": "(photon:3.0) openssl-perl"
},
{
"AffectedVersionList": [
"version<1:1.1.1g-11.tl3"
],
"FixedVersionList": [
"version>=1:1.1.1g-11.tl3"
],
"Name": "(tlinux:3) openssl"
},
{
"AffectedVersionList": [
"version<1:1.1.1g-11.tl3"
],
"FixedVersionList": [
"version>=1:1.1.1g-11.tl3"
],
"Name": "(tlinux:3) openssl-devel"
},
{
"AffectedVersionList": [
"version<1:1.1.1g-11.tl3"
],
"FixedVersionList": [
"version>=1:1.1.1g-11.tl3"
],
"Name": "(tlinux:3) openssl-libs"
},
{
"AffectedVersionList": [
"version<1:1.1.1g-11.tl3"
],
"FixedVersionList": [
"version>=1:1.1.1g-11.tl3"
```

```
],
"Name": "(tlinux:3) openssl-perl"
},
{
"AffectedVersionList": [
"version<0:1.0.2j-60.60"
],
"FixedVersionList": [
"version>=0:1.0.2j-60.60"
],
"Name": "(sles:7) libopenssl-devel"
},
{
"AffectedVersionList": [
"version<0:1.0.2j-60.60"
],
"FixedVersionList": [
"version>=0:1.0.2j-60.60"
],
"Name": "(sles:7) libopenssl1_0_0"
},
{
"AffectedVersionList": [
"version<0:1.0.2j-60.60"
],
"FixedVersionList": [
"version>=0:1.0.2j-60.60"
],
"Name": "(sles:7) libopenssl1_0_0-32bit"
},
{
"AffectedVersionList": [
"version<0:1.0.2j-60.60"
],
"FixedVersionList": [
"version>=0:1.0.2j-60.60"
],
"Name": "(sles:7) libopenssl1_0_0-hmac"
},
{
"AffectedVersionList": [
"version<0:1.0.2j-60.60"
],
"FixedVersionList": [
"version>=0:1.0.2j-60.60"
],
"Name": "(sles:7) libopenssl1_0_0-hmac-32bit"
},
{
"AffectedVersionList": [
"version<0:1.0.2j-60.60"
],
"FixedVersionList": [
"version>=0:1.0.2j-60.60"
],
```

```
"Name": "(sles:7) openssl"
},
{
"AffectedVersionList": [
"version<0:1.0.2j-60.60"
],
"FixedVersionList": [
"version>=0:1.0.2j-60.60"
],
"Name": "(sles:7) openssl-doc"
},
{
"AffectedVersionList": [
"version<1:1.1.1g-11.el8"
],
"FixedVersionList": [
"version>=1:1.1.1g-11.el8"
],
"Name": "(centos:8) openssl"
},
{
"AffectedVersionList": [
"version<1:1.1.1g-11.el8"
],
"FixedVersionList": [
"version>=1:1.1.1g-11.el8"
],
"Name": "(centos:8) openssl-devel"
},
{
"AffectedVersionList": [
"version<1:1.1.1g-11.el8"
],
"FixedVersionList": [
"version>=1:1.1.1g-11.el8"
],
"Name": "(centos:8) openssl-libs"
},
{
"AffectedVersionList": [
"version<1:1.1.1g-11.el8"
],
"FixedVersionList": [
"version>=1:1.1.1g-11.el8"
],
"Name": "(centos:8) openssl-perl"
},
{
"AffectedVersionList": [
"version<0:1.0.2j-60.60"
],
"FixedVersionList": [
"version>=0:1.0.2j-60.60"
],
"Name": "(sles:8) libopenssl-devel"
```

```
},
{
"AffectedVersionList": [
"version<0:1.0.2j-60.60"
],
"FixedVersionList": [
"version>=0:1.0.2j-60.60"
],
"Name": "(sles:8) libopenssl1_0_0"
},
{
"AffectedVersionList": [
"version<0:1.0.2j-60.60"
],
"FixedVersionList": [
"version>=0:1.0.2j-60.60"
],
"Name": "(sles:8) libopenssl1_0_0-32bit"
},
{
"AffectedVersionList": [
"version<0:1.0.2j-60.60"
],
"FixedVersionList": [
"version>=0:1.0.2j-60.60"
],
"Name": "(sles:8) libopenssl1_0_0-hmac"
},
{
"AffectedVersionList": [
"version<0:1.0.2j-60.60"
],
"FixedVersionList": [
"version>=0:1.0.2j-60.60"
],
"Name": "(sles:8) libopenssl1_0_0-hmac-32bit"
},
{
"AffectedVersionList": [
"version<0:1.0.2j-60.60"
],
"FixedVersionList": [
"version>=0:1.0.2j-60.60"
],
"Name": "(sles:8) openssl"
},
{
"AffectedVersionList": [
"version<0:1.0.2j-60.60"
],
"FixedVersionList": [
"version>=0:1.0.2j-60.60"
],
"Name": "(sles:8) openssl-doc"
},
```

```
{
"AffectedVersionList": [
"version<1.1.0l-1~deb9u5"
],
"FixedVersionList": [
"version>=1.1.0l-1~deb9u5"
],
"Name": "(debian:9) openssl"
},
{
"AffectedVersionList": [
"version<1.0.2u-1~deb9u1"
],
"FixedVersionList": [
"version>=1.0.2u-1~deb9u1"
],
"Name": "(debian:9) openssl1.0"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-3.14"
],
"FixedVersionList": [
"version>=0:1.0.2p-3.14"
],
"Name": "(sles:9) libopenssl-1_0_0-devel"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-3.14"
],
"FixedVersionList": [
"version>=0:1.0.2p-3.14"
],
"Name": "(sles:9) libopenssl1_0_0"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-3.14"
],
"FixedVersionList": [
"version>=0:1.0.2p-3.14"
],
"Name": "(sles:9) libopenssl1_0_0-32bit"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-3.14"
],
"FixedVersionList": [
"version>=0:1.0.2p-3.14"
],
"Name": "(sles:9) libopenssl1_0_0-hmac"
},
{
```

```
"AffectedVersionList": [
"version<0:1.0.2p-3.14"
],
"FixedVersionList": [
"version>=0:1.0.2p-3.14"
],
"Name": "(sles:9) libopenssl1_0_0-hmac-32bit"
},
{
"AffectedVersionList": [
"version<0:1.1.1d-2.20"
],
"FixedVersionList": [
"version>=0:1.1.1d-2.20"
],
"Name": "(sles:9) libopenssl1_1"
},
{
"AffectedVersionList": [
"version<0:1.1.1d-2.20"
],
"FixedVersionList": [
"version>=0:1.1.1d-2.20"
],
"Name": "(sles:9) libopenssl1_1-32bit"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-3.14"
],
"FixedVersionList": [
"version>=0:1.0.2p-3.14"
],
"Name": "(sles:9) openssl-1_0_0"
},
{
"AffectedVersionList": [
"version<0:1.0.2p-3.14"
],
"FixedVersionList": [
"version>=0:1.0.2p-3.14"
],
"Name": "(sles:9) openssl-1_0_0-doc"
},
{
"AffectedVersionList": [
"version<0:1.1.1d-2.20"
],
"FixedVersionList": [
"version>=0:1.1.1d-2.20"
],
"Name": "(sles:9) openssl-1_1"
},
{
"AffectedVersionList": [
```

```
"version<1.1.1e-1"
],
"FixedVersionList": [
"version>=1.1.1e-1"
],
"Name": "(debian:unstable) openssl"
},
{
"AffectedVersionList": [
"version<1.1.1d-r2"
],
"FixedVersionList": [
"version>=1.1.1d-r2"
],
"Name": "(alpine:v3.10) openssl"
},
{
"AffectedVersionList": [
"version<1.1.1d-r3"
],
"FixedVersionList": [
"version>=1.1.1d-r3"
],
"Name": "(alpine:v3.11) openssl"
},
{
"AffectedVersionList": [
"version<1.1.1d-r3"
],
"FixedVersionList": [
"version>=1.1.1d-r3"
],
"Name": "(alpine:v3.12) openssl"
},
{
"AffectedVersionList": [
"version<1.1.1d-r3"
],
"FixedVersionList": [
"version>=1.1.1d-r3"
],
"Name": "(alpine:v3.13) openssl"
},
{
"AffectedVersionList": [
"version<1.1.1d-r3"
],
"FixedVersionList": [
"version>=1.1.1d-r3"
],
"Name": "(alpine:v3.14) openssl"
},
{
"AffectedVersionList": [
"version<1.1.1d-r3"
```

```
],
"FixedVersionList": [
"version>=1.1.1d-r3"
],
"Name": "(alpine:v3.15) openssl"
},
{
"AffectedVersionList": [
"version<1.1.1d-r3"
],
"FixedVersionList": [
"version>=1.1.1d-r3"
],
"Name": "(alpine:v3.15) openssl3"
},
{
"AffectedVersionList": [
"version<1.1.1d-r3"
],
"FixedVersionList": [
"version>=1.1.1d-r3"
],
"Name": "(alpine:v3.16) openssl"
},
{
"AffectedVersionList": [
"version<1.1.1d-r3"
],
"FixedVersionList": [
"version>=1.1.1d-r3"
],
"Name": "(alpine:v3.16) openssl3"
},
{
"AffectedVersionList": [
"version<1.1.1d-r3"
],
"FixedVersionList": [
"version>=1.1.1d-r3"
],
"Name": "(alpine:v3.17) openssl"
},
{
"AffectedVersionList": [
"version<1.0.2u-r0"
],
"FixedVersionList": [
"version>=1.0.2u-r0"
],
"Name": "(alpine:v3.8) openssl"
},
{
"AffectedVersionList": [
"version<1.1.1d-r2"
],
```

```
"FixedVersionList": [
"version>=1.1.1d-r2"
],
"Name": "(alpine:v3.9) openssl"
}
],
"CVSSv2Info": {
"AccessComplexity": "LOW",
"AccessVector": "NETWORK",
"Authentication": "NONE",
"AvailabilityImpact": "NONE",
"CVSS": 5,
"ConImpact": "PARTIAL",
"IntegrityImpact": "NONE"
},
"CVSSv2Vector": "(AV:N/AC:L/Au:N/C:P/I:N/A:N)",
"CVSSv3Info": {
"AttackComplexity": "LOW",
"AttackVector": "NETWORK",
"AvailabilityImpact": "NONE",
"CVSS": 5.3,
"ConImpact": "LOW",
"IntegrityImpact": "NONE",
"PrivilegesRequired": "NONE",
"Scope": "UNCHANGED",
"UserInteraction": "NONE"
},
"CVSSv3Vector": "CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N",
"CWEID": "CWE-200",
"Category": "整数溢出",
"CategoryType": "应用漏洞",
"DefenseSolution": "目前厂商已发布升级补丁以修复漏洞，补丁获取链接：https://www.openssl.org/news/secadv/20191206.txt",
"Description": "OpenSSL是OpenSSL团队的一个开源的能够实现安全套接层（SSLv2/v3）和安全传输层（TLSv1）协议的通用加密库。该产品支持多种加密算法，包括对称密码、哈希算法、安全散列算法等。OpenSSL1.1.1版本至1.1.1d版本和1.0.2版本至1.0.2t版本中存在信息泄露漏洞。该漏洞源于网络系统或产品在运行过程中存在配置等错误。未授权的攻击者可利用漏洞获取受影响组件敏感信息。",
"OfficialSolution": "",
"ReferenceList": [
"http://lists.opensuse.org/opensuse-security-announce/2020-01/msg00030.html",
"http://packetstormsecurity.com/files/155754/Slackware-Security-Advisory-openssl-Updates.html",
"https://git.openssl.org/gitweb/?p=openssl.git;a=commitdiff;h=419102400a2811582a7a3d4a4e317d72e5ce0a8f",
"https://git.openssl.org/gitweb/?p=openssl.git;a=commitdiff;h=f1c5eea8a817075d31e43f5876993c6710238c98",
"https://lists.fedoraproject.org/archives/list/package-announce@lists.fedoraproject.org/message/XVEP3LAK4JSPRXFO4QF4GG2IVXADV3SO/",
"https://seclists.org/bugtraq/2019/Dec/39",
"https://seclists.org/bugtraq/2019/Dec/46",
"https://security.gentoo.org/glsa/202004-10",
"https://security.netapp.com/advisory/ntap-20191210-0001/",
"https://www.debian.org/security/2019/dsa-4594",
"https://www.openssl.org/news/secadv/20191206.txt",
"https://www.tenable.com/security/tns-2019-09"
],
"SubmitTime": "2019-12-07 02:15:00"
},
"Summary": {
"CNNVDID": "CNNVD-201912-272",
```

```
"CNVDID": "CNVD-2020-03864",
"CVEID": "CVE-2019-1551",
"IsSuggest": false,
"Name": "OpenSSL 信息泄露漏洞（CVE-2019-1551）",
"Severity": "Medium",
"VulID": "pcmgr-209550"
}
}
]
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.AccountNotEnough	账户流量余额不足。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
MissingParameter	缺少参数错误。

查询组件的版本列表

最近更新时间：2024-10-29 21:02:27

1. 接口描述

接口请求域名： bsca.tencentcloudapi.com 。

查询特定组件的版本列表

默认接口请求频率限制：10000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：DescribeKBComponentVersionList。
Version	是	String	公共参数，本接口取值：2021-08-11。
Region	否	String	公共参数，此参数为可选参数。
PURL	是	PURL	要查询的组件 PURL
PageNumber	否	Integer	页号 示例值：1
PageSize	否	Integer	页大小 示例值：10
Order	否	String	排序方式，可以是"ASC"或"DESC"，默认"DESC" 示例值：ASC
OrderBy.N	否	Array of String	排序字段，可能的字段包括“Version”、"PublishTime" 示例值：["PublishTime","Version"]
Filter	否	ComponentTagFilter	Tag筛选

3. 输出参数

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

4. 示例

示例1 查询知识库组件的版本列表

查询ristretto组件的版本列表

输入示例

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

{
"PURL": {
"Name": "ristretto",
"Protocol": "golang",
"Namespace": "",
"Subpath": "",
"Version": ""
}
}
```

输出示例

```
{
"Response": {
"VersionList": [
{
"LicenseExpression": "gpl-3.0",
"PURL": {
"Name": "xxl-job-core",
"Namespace": "com.xuxueli",
"Protocol": "maven",
"Qualifiers": [],
"Subpath": "",
"Version": "2.4.1"
},
"VersionInfo": {
"CopyrightList": [],
"PublishTime": "2024-04-17 08:50:11.000000",
"TagList": [
"ContainsVulnerability"
]
}
}
],
"RequestId": "101705d6-b972-4dbf-9d10-0b075d509d3d"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.AccountNotEnough	账户流量余额不足。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
MissingParameter	缺少参数错误。
ResourceNotFound	资源不存在。

数据结构

最近更新时间：2024-10-29 21:02:28

AffectedComponent

受漏洞影响的组件信息。

被如下接口引用：DescribeKBVulnerability。

名称	类型	描述
Name	String	受漏洞影响的组件名称 注意：此字段可能返回 null，表示取不到有效值。 示例值：apache-log4j
AffectedVersionList	Array of String	受漏洞影响的版本 注意：此字段可能返回 null，表示取不到有效值。 示例值：["2.4.0<=version<2.12.2"]
FixedVersionList	Array of String	修复此漏洞的版本 注意：此字段可能返回 null，表示取不到有效值。 示例值：["2.15.0"]

CVSSV2Info

CVSSv2.0详细信息。

被如下接口引用：DescribeKBVulnerability。

名称	类型	描述
CVSS	Float	CVE评分。 示例值：6.8
AccessVector	String	AccessVector 攻击途径。 取值范围： • NETWORK 远程 • ADJACENT_NETWORK 近邻 • LOCAL 本地 示例值：LOCAL
AccessComplexity	String	AccessComplexity 攻击复杂度。 取值范围： • HIGH 高 • MEDIUM 中 • LOW 低 示例值：HIGH
Authentication	String	Authentication 身份验证。 取值范围： • MULTIPLE 多系统认证 • SINGLE 单系统认证 • NONE 无 示例值：MULTIPLE
ConImpact	String	ConfidentialityImpact 机密性影响。 取值范围： • NONE 无 • PARTIAL 部分 • COMPLETE 完整 示例值：PARTIAL

名称	类型	描述
IntegrityImpact	String	IntegrityImpact 完整性影响。 取值范围： • NONE 无 • PARTIAL 部分 • COMPLETE 完整 示例值：PARTIAL
AvailabilityImpact	String	AvailabilityImpact 可用性影响。 取值范围： • NONE 无 • PARTIAL 部分 • COMPLETE 完整 示例值：PARTIAL

CVSSV3Info

Cvssv3.0详细信息。

被如下接口引用：DescribeKBVulnerability。

名称	类型	描述
CVSS	Float	CVE评分。 示例值：5.5
AttackVector	String	AttackVector 攻击途径。 取值范围： • NETWORK 远程 • ADJACENT_NETWORK 近邻 • LOCAL 本地 • PHYSICAL 物理 示例值：NETWORK
AttackComplexity	String	AttackComplexity 攻击复杂度。 取值范围： • HIGH 高 • LOW 低 示例值：HIGH
PrivilegesRequired	String	PrivilegesRequired 触发特权。 取值范围： • HIGH 高 • LOW 低 • NONE 无 示例值：HIGH
UserInteraction	String	UserInteraction 交互必要性。 取值范围： • NONE 无 • REQUIRED 需要 示例值：REQUIRED
Scope	String	Scope 绕过安全边界。 取值范围： • UNCHANGED 否 • CHANGED 能 示例值：UNCHANGED

名称	类型	描述
ConImpact	String	ConfidentialityImpact 机密性影响。 取值范围： • NONE 无 • LOW 低 • HIGH 高 示例值：LOW
IntegrityImpact	String	IntegrityImpact 完整性影响。 取值范围： • NONE 无 • LOW 低 • HIGH 高 示例值：LOW
AvailabilityImpact	String	AvailabilityImpact 可用性影响。 取值范围： • NONE 无 • LOW 低 • HIGH 高 示例值：LOW

Component

描述一个第三方组件的源信息。

被如下接口引用：DescribeKBComponent, SearchKBComponent。

名称	类型	描述
PURL	PURL	第三方组件的PURL
Homepage	String	第三方组件的主页 示例值：https://www.openssl.org
Summary	String	第三方组件的简介 示例值：OpenSSL is a toolkit for supporting cryptography.
NicknameList	Array of String	第三方组件的别名列表 注意：此字段可能返回 null，表示取不到有效值。 示例值：["gmock", "googlemock", "gmock-gtest-all"]
CodeLocationList	Array of String	第三方组件的代码位置列表 注意：此字段可能返回 null，表示取不到有效值。 示例值： ["https://github.com/google/googletest.git","https://github.com/abseil/googletest.git"]
LicenseExpression	String	第三方组件的许可证表达式 示例值：apache-2.0
VersionInfo	ComponentVersionInfo	第三方组件的版本信息(如果匹配到版本) 注意：此字段可能返回 null，表示取不到有效值。
LastUpdateTime	String	第三方组件的最后更新时间 注意：此字段可能返回 null，表示取不到有效值。 示例值：2024-10-03T11:12:40Z
TagList	Array of String	第三方组件的类型标签 注意：此字段可能返回 null，表示取不到有效值。 示例值：["hardware"]

ComponentTagFilter

筛选条件，同一个Tag不能同时出现在IncludeTags和ExcludeTags，可能的Tag包括："CopyrightUpdated", "LicenseUpdated", "ContainsVulnerability"

被如下接口引用：DescribeKBComponentVersionList。

名称	类型	必选	描述
IncludeTags	Array of String	否	包括的Tag 示例值：["CopyrightUpdated"]
ExcludeTags	Array of String	否	排除的Tag 示例值：["LicenseUpdated"]

ComponentVersion

描述一个组件版本。

被如下接口引用：DescribeKBComponentVersionList。

名称	类型	描述
PURL	PURL	该组件的PURL 注意：此字段可能返回 null，表示取不到有效值。
LicenseExpression	String	该组件版本的许可证表达式 注意：此字段可能返回 null，表示取不到有效值。 示例值：mit
VersionInfo	ComponentVersionInfo	组件的版本信息 注意：此字段可能返回 null，表示取不到有效值。

ComponentVersionInfo

描述组件版本的详情，包含组件发布时间、Copyright列表、组件描述Tag。

被如下接口引用：DescribeKBComponent, DescribeKBComponentVersionList, SearchKBComponent。

名称	类型	描述
PublishTime	String	版本发布时间 注意：此字段可能返回 null，表示取不到有效值。 示例值：2024-04-17 08:50:11.000000
CopyrightList	Array of String	当前版本的所有copyright 注意：此字段可能返回 null，表示取不到有效值。 示例值：["Copyright (c) 1998-2018 The OpenSSL Project", "Copyright (c) 1995-1998 Eric Young (eay@cryptsoft.com)", "holder is Tim Hudson (tjh@cryptsoft.com)"]
TagList	Array of String	版本标签 注意：此字段可能返回 null，表示取不到有效值。 示例值：["ContainsVulnerability", "LicenseUpdated"]

ComponentVulnerabilitySummary

与输入组件相关的漏洞信息摘要信息。

被如下接口引用：DescribeKBComponentVulnerability。

名称	类型	描述
PURL	PURL	用于匹配漏洞的PURL 注意：此字段可能返回 null，表示取不到有效值。
CanBeFixed	Boolean	该组件是否包含修复漏洞的官方补丁 示例值：true

名称	类型	描述
FixedVersion	String	修复漏洞的组件版本号 示例值：>1.1.1c
AffectedVersion	String	漏洞影响的组件版本号 示例值：1.1.1
AffectedComponent	String	漏洞影响组件 示例值：openssl
RiskLevel	String	漏洞在该产品中的风险等级 • Critical • High • Medium • Low 示例值：Medium

ComponentVulnerabilityUnion

描述组件漏洞相关概览信息。

被如下接口引用：DescribeKBComponentVulnerability。

名称	类型	描述
Summary	VulnerabilitySummary	漏洞概览信息
SummaryInComponent	ComponentVulnerabilitySummary	与组件相关的漏洞概览信息

LicenseDetail

描述许可证的详细信息。

被如下接口引用：DescribeKBLicense。

名称	类型	描述
Content	String	许可证内容 示例值： TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION...
ConditionSet	Array of LicenseRestriction	许可证允许信息列表
ForbiddenSet	Array of LicenseRestriction	许可证要求信息列表
PermissionSet	Array of LicenseRestriction	许可证禁止信息列表

LicenseRestriction

License约束信息。

被如下接口引用：DescribeKBLicense。

名称	类型	描述
Name	String	license约束的名称。 示例值：Commercial Use
Description	String	license约束的描述。 示例值：The licensed material

LicenseSummary

描述许可证的概览信息。

被如下接口引用：DescribeKBLicense。

名称	类型	必选	描述
Key	String	是	许可证标识符 示例值：apache-2.0
SPDXKey	String	是	许可证的SPDX标识符，见 https://spdx.org/licenses/ 示例值：Apache-2.0
ShortName	String	是	许可证短名称 示例值：Apache 2.0
Name	String	是	许可证完整名称 示例值：Apache License 2.0
Risk	String	是	License风险等级 • NotDefined • LowRisk • MediumRisk • HighRisk 示例值：LowRisk
Source	String	是	许可证来源URL 示例值：https://spdx.org/licenses/Apache-2.0.html

LicenseUnion

许可证详细信息。

被如下接口引用：DescribeKBLicense。

名称	类型	描述
LicenseSummary	LicenseSummary	许可证概览信息
LicenseDetail	LicenseDetail	许可证详细信息

PURL

PURL(Package URL)用于定位一个产品或组件，见 https://github.com/package-url/purl-spec。

被如下接口引用：DescribeKBComponent, DescribeKBComponentVersionList, DescribeKBComponentVulnerability, MatchKBPURLList, SearchKBComponent。

名称	类型	必选	描述
Name	String	是	组件名称 示例值：curl
Protocol	String	否	组件所属的类型，如：github, gitlab, generic, deb, rpm, maven 等 示例值：deb
Namespace	String	否	组件名的前缀名，如github和gitlab的用户名，deb的操作系统，maven包的group id等 示例值：debian
Qualifiers	Array of Qualifier	否	修饰组件的额外属性 注意：此字段可能返回 null，表示取不到有效值。
Subpath	String	否	相对于组件包根位置的子目录 示例值：/release

名称	类型	必选	描述
Version	String	否	组件版本号 示例值：7.50.3

Qualifier

PURL下的Qualifier属性类型，用于定义第三方组件的额外属性，见 https://github.com/package-url/purl-spec。

被如下接口引用：DescribeKBComponent, DescribeKBComponentVersionList, DescribeKBComponentVulnerability, MatchKBPURLList。

名称	类型	必选	描述
Key	String	是	额外属性的名称。 示例值：distro_version
Value	String	是	额外属性的值。 示例值：14.04

VulnerabilityDetail

描述漏洞详细信息。

被如下接口引用：DescribeKBVulnerability。

名称	类型	描述
Category	String	漏洞类别 示例值：反序列化
CategoryType	String	漏洞分类 示例值：web应用漏洞
Description	String	漏洞描述 示例值：ApacheLog4j是美国阿帕奇（Apache）基金会的一款基于Java的开源日志记录工具。ApacheLog4J存在输入验证错误漏洞.Log4j-2中存在JNDI注入漏洞，当程序将用户输入的数据进行日志记录时，即可触发此漏洞，成功利用此漏洞可以在目标服务器上执行任意代码。
OfficialSolution	String	漏洞官方解决方案 示例值：升级到最新无漏洞版本
ReferenceList	Array of String	漏洞信息参考列表 示例值：["https://logging.apache.org/log4j/2.x/security.html", "http://www.openwall.com/lists/oss-security/2021/12/10/1"]
DefenseSolution	String	漏洞防御方案 示例值：目前厂商已发布升级补丁以修复漏洞，补丁获取链接：https://www.openssl.org/news/secadv/2019
CVSSv2Info	CVSSV2Info	漏洞CVSSv2信息 注意：此字段可能返回 null，表示取不到有效值。
CVSSv3Info	CVSSV3Info	漏洞CVSSv3信息 注意：此字段可能返回 null，表示取不到有效值。
SubmitTime	String	漏洞提交时间 示例值：2019-09-11 01:15:00
UpdateTime	String	漏洞更新时间 示例值：2019-09-11 01:15:00
CWEID	String	CWE编号 示例值：CWE-311
CVSSv2Vector	String	漏洞CVSSv2向量 示例值：(AV:L/AC:M/Au:N/C:P/I:N/A:N)

名称	类型	描述
CVSSv3Vector	String	漏洞CVSSv3向量 示例值：CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:H/I:N/A:N
AffectedComponentList	Array of AffectedComponent	漏洞影响的组件列表，仅当查询单个漏洞时有效

VulnerabilitySummary

描述漏洞的摘要信息。

被如下接口引用：DescribeKBComponentVulnerability, DescribeKBVulnerability。

名称	类型	描述
VulID	String	漏洞ID 示例值：pcmgr-303514
CVEID	String	漏洞所属CVE编号 示例值：CVE-2019-11236
CNVDID	String	漏洞所属CNVD编号 示例值：CNVD-2021-100238
CNNVDID	String	漏洞所属CNNVD编号 示例值：CNNVD-202112-799
Name	String	漏洞名称 示例值：OpenSSL 拒绝服务漏洞
IsSuggest	Boolean	该漏洞是否是需重点关注的漏洞 示例值：false
Severity	String	漏洞风险等级 • Critical • High • Medium • Low 示例值：Low
ArchitectureList	Array of String	架构信息，如x86、ARM等 注意：此字段可能返回 null，表示取不到有效值。 示例值：["x86_64"]
PatchUrlList	Array of String	patch链接 注意：此字段可能返回 null，表示取不到有效值。 示例值：["https://github.com/apache/logging-log4j2/commit/c77b3cb39312b83b053d23a2158b99ac7de44dd3"]

VulnerabilityUnion

描述漏洞的详细信息。

被如下接口引用：DescribeKBVulnerability。

名称	类型	描述
Summary	VulnerabilitySummary	漏洞概览信息
Detail	VulnerabilityDetail	漏洞详细信息

错误码

最近更新时间：2022-10-18 06:12:17

功能说明

如果返回结果中存在 Error 字段，则表示调用 API 接口失败。例如：

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

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

错误码列表

公共错误码

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

错误码	说明
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
FailedOperation.AccountNotEnough	账户流量余额不足。