AI 临床助手

API 文档

腾讯云

文档目录

API 文档
更新历史

最近更新时间：2024-12-24 09:13:40

第 1 次发布

发布时间：2024-12-24 09:13:25

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- GetDrugIndications
- LoginHisTool
- LoginOutHisTool
- SmartDrugInfo
- SmartPredict
- SyncDepartment
- SyncStandardDict
- UploadDrugs

新增数据结构：

- Abnormals
- CommonHeader
- CriticalInfo
- CurrentDiagnosis
- Department
- DiagnosisInfo
- Dict
- DifferDiagnosis
- DocInfo
- DoctorInfo
- Drug
- DrugList
- DrugUsage
- DurgPropertyInfo
- EmrDiagnosises
- EmrQuality
- GetDrugIndicationsReqData
- GetDrugIndicationsResp
- HealthPrescriptions
- IndicationsDrug
- LoginData
- LoginDataResp
- LoginHeader
- LoginOutData
- LoginOutHeader
- LoginOutResponseData
- OperateResp
- PatientBaseInfo
- PatientFamilyHistory
- PatientHistory
- PhysicalExam
- RationalDrugInfo
- RecommendedUsage

- RecordQuality
- ReferralInfo
- RequestCase
- RiskInfo
- SmartDrugInfoData
- SmartDrugInfoResp
- SmartPredictReqData
- SmartPredictRespData
- SuspectedDiagnosis
- SyncDepartmentData
- SyncDepartmentRespData
- SyncDictData
- TreatmentGuide
- UploadDrugData
- VitalSignsInfo

简介

最近更新时间：2024-12-24 09:13:38

欢迎使用 AI 临床助手 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

AI 临床助手（AI Clinical Assistant, ACA）是临床医生的智能帮手。AI 临床助手基于医学人工智能、医学知识图谱等先进技术，考虑医生在病史采集、下诊断、制定治疗方案、开具处方、医生教育等多种应用场景下的特性需求，依据客观权威的知识来源，协助提升医疗服务效率。

API 概览

最近更新时间：2024-12-24 09:13:40

AI 临床助手相关接口

接口名称	接口功能	频率限制（次/秒）
GetDrugIndications	药品适应症接口	20
LoginHisTool	登录获取辅诊访问Token	20
LoginOutHisTool	登出his工具	20
SmartDrugInfo	智能用药接口	20
SmartPredict	智能预测	20
SyncDepartment	科室同步接口	20
SyncStandardDict	同步标准字典接口	20
UploadDrugs	药品同步	20

⚠ 注意：

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

调用方式
请求结构

最近更新时间：2024-12-24 09:13:38

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 aca.tencentcloudapi.com ，也支持指定地域域名访问，例如广州地域的域名为 aca.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 aca.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）	aca.tencentcloudapi.com
华南地区(广州)	aca.ap-guangzhou.tencentcloudapi.com
华东地区(上海)	aca.ap-shanghai.tencentcloudapi.com
华北地区(北京)	aca.ap-beijing.tencentcloudapi.com
西南地区(成都)	aca.ap-chengdu.tencentcloudapi.com
西南地区(重庆)	aca.ap-chongqing.tencentcloudapi.com
港澳台地区(中国香港)	aca.ap-hongkong.tencentcloudapi.com
亚太东南(新加坡)	aca.ap-singapore.tencentcloudapi.com
亚太东南(曼谷)	aca.ap-bangkok.tencentcloudapi.com
亚太南部(孟买)	aca.ap-mumbai.tencentcloudapi.com
亚太东北(首尔)	aca.ap-seoul.tencentcloudapi.com
亚太东北(东京)	aca.ap-tokyo.tencentcloudapi.com
美国东部(弗吉尼亚)	aca.na-ashburn.tencentcloudapi.com
美国西部(硅谷)	aca.na-siliconvalley.tencentcloudapi.com
欧洲地区(法兰克福)	aca.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：2024-12-24 09:13:39

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

公共参数的具体内容会因您使用的签名方法版本不同而有所差异。

使用签名方法 v3 的公共参数

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

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

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

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

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

HTTP GET 请求结构示例：

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

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

HTTP POST（application/json）请求结构示例：

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

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

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

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

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

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

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

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

10
--58731222010402--
```

使用签名方法 v1 的公共参数

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

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

参数名称	类型	必选	描述
Token	String	否	即 安全凭证服务 所颁发的临时安全凭证中的 Token，使用时需要将 SecretId 和 SecretKey 的值替换为临时安全凭证中的 TmpSecretId 和 TmpSecretKey。使用长期密钥时不能设置此 Token 字段。
Language	String	否	指定接口返回的语言，仅部分接口支持此参数。取值：zh-CN，en-US。zh-CN 返回中文，en-US 返回英文。

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

HTTP GET 请求结构示例：

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

Host: cvm.tencentcloudapi.com
```

HTTP POST 请求结构示例：

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

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

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

地域列表

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

地域	取值
华南地区（广州）	ap-guangzhou

签名方法 v3

最近更新时间：2024-12-24 09:13:39

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

推荐使用 API Explorer　　点击调试

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

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

为什么要进行签名

签名通过以下方式帮助保护请求：

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

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

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

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

申请安全凭证

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

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

申请安全凭证的具体步骤如下：

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

签名版本 v3 签名过程

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

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

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

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

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

```
curl -X POST https://cvm.tencentcloudapi.com \
-H "Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host;x-tc-action, Signature=e6ff5806def0c98fc206796a81228cc6cf8dc4ecaced4b7cc1bc7b5fb1578df0" \
-H "Content-Type: application/json; charset=utf-8" \
-H "Host: cvm.tencentcloudapi.com" \
-H "X-TC-Action: DescribeInstances" \
-H "X-TC-Timestamp: 1551113065" \
-H "X-TC-Version: 2017-03-12" \
-H "X-TC-Region: ap-guangzhou" \
-d '{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}'
```

下面详细解释签名计算过程。

1. 拼接规范请求串

按如下伪代码格式拼接规范请求串（CanonicalRequest）：

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

字段名称	解释
HTTPRequestMethod	HTTP 请求方法（GET、POST）。此示例取值为 `POST`。
CanonicalURI	URI 参数，API 3.0 固定为正斜杠（/）。
CanonicalQueryString	发起 HTTP 请求 URL 中的查询字符串，对于 POST 请求，固定为空字符串""，对于 GET 请求，则为 URL 中问号（?）后面的字符串内容，例如：Limit=10&Offset=0。 注意：CanonicalQueryString 需要参考 RFC3986 进行 URLEncode 编码（特殊字符编码后需大写字母），字符集 UTF-8。推荐使用编程语言标准库进行编码。
CanonicalHeaders	参与签名的头部信息，至少包含 host 和 content-type 两个头部，也可加入其他头部参与签名以提高自身请求的唯一性和安全性，此示例额外增加了接口名头部。 拼接规则： 1. 头部 key 和 value 统一转成小写，并去掉首尾空格，按照 key:value\n 格式拼接； 2. 多个头部，按照头部 key（小写）的 ASCII 升序进行拼接。 此示例计算结果是 `content-type:application/json; charset=utf-8\nhost:cvm.tencentcloudapi.com\nx-tc-action:describeinstances\n`。 注意：content-type 必须和实际发送的相符合，有些编程语言网络库即使未指定也会自动添加 charset 值，如果签名时和发送时不一致，服务器会返回签名校验失败。
SignedHeaders	参与签名的头部信息，说明此次请求有哪些头部参与了签名，和 CanonicalHeaders 包含的头部内容是一一对应的。content-type 和 host 为必选头部。 拼接规则： 1. 头部 key 统一转成小写； 2. 多个头部 key（小写）按照 ASCII 升序进行拼接，并且以分号（;）分隔。 此示例为 content-type;host;x-tc-action
HashedRequestPayload	请求正文（payload，即 body，此示例为 `{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}`）的哈希值，计算伪代码为 Lowercase(HexEncode(Hash.SHA256(RequestPayload)))，即对 HTTP 请求正文做 SHA256 哈希，然后十六进制编

字段名称	解释
	码，最后编码串转换成小写字母。对于 GET 请求，RequestPayload 固定为空字符串。此示例计算结果是 35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064。

根据以上规则，示例中得到的规范请求串如下：

```
POST
/

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

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

2. 拼接待签名字符串

按如下格式拼接待签名字符串：

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

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

注意：

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

根据以上规则，示例中得到的待签名字符串如下：

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

3. 计算签名

1）计算派生签名密钥，伪代码如下：

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

派生出的密钥 `SecretDate`、`SecretService` 和 `SecretSigning` 是二进制的数据，可能包含不可打印字符，将其转为十六进制字符串打印的输出分别为：
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 `********************************`。
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 `e6ff5806def0c98fc206796a81228cc6cf8dc4ecaced4b7cc1bc7b5fb1578df0`。

4. 拼接 Authorization

按如下格式拼接 Authorization：

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

字段名称	解释
Algorithm	签名方法，固定为 `TC3-HMAC-SHA256`。
SecretId	密钥对中的 SecretId，即 `AKID********************************`。
CredentialScope	见上文，凭证范围。此示例计算结果是 `2019-02-25/cvm/tc3_request`。
SignedHeaders	见上文，参与签名的头部信息。此示例取值为 `content-type;host;x-tc-action`。
Signature	签名值。此示例计算结果是 `e6ff5806def0c98fc206796a81228cc6cf8dc4ecaced4b7cc1bc7b5fb1578df0`。

根据以上规则，示例中得到的值为：

```
TC3-HMAC-SHA256 Credential=AKID********************************/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;hos
t;x-tc-action, Signature=e6ff5806def0c98fc206796a81228cc6cf8dc4ecaced4b7cc1bc7b5fb1578df0
```

最终完整的调用信息如下：

```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKID*******************************/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host;x-tc-action, Signature=e6ff5806def0c98fc206796a81228cc6cf8dc4ecaced4b7cc1bc7b5fb1578df0
Content-Type: application/json; charset=utf-8
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1551113065
X-TC-Region: ap-guangzhou

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

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

签名演示

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

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

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

- Signature Demo

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

Java

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

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

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

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

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

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

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

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

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

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

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

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

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

Python

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

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

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

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

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

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


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

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

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

Golang

```
package main

import (
"crypto/hmac"
```

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

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

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

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

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

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

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

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

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

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

PHP

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

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

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

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

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

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

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

Ruby

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

```ruby
require 'json'
require 'time'
require 'openssl'

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

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

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

puts canonical_request

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

# ************* 步骤 3：计算签名 *************
digest = OpenSSL::Digest.new('sha256')
secret_date = OpenSSL::HMAC.digest(digest, 'TC3' + secret_key, date)
```

```
secret_service = OpenSSL::HMAC.digest(digest, secret_date, service)
secret_signing = OpenSSL::HMAC.digest(digest, secret_service, 'tc3_request')
signature = OpenSSL::HMAC.hexdigest(digest, secret_signing, string_to_sign)
puts signature

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

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

DotNet

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

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

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

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

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

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

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

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

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

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

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

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

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

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

NodeJS

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

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

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

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

function main(){
```

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

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

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

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

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

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

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

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

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

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

C++

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

using namespace std;

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

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

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

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

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

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

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

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

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

return (ss.str());
}

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

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

```
}
return output;
}

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

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

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

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

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

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

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

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

C

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

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

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

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

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

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


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

}

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

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

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

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

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

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

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


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

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

char signature[128] = {0};
```

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

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

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

其他语言

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

签名失败

存在以下签名失败的错误码，请根据实际情况处理。

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

签名方法

最近更新时间：2024-12-24 09:13:39

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

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

推荐使用 API Explorer　　点击调试

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

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

1. 申请安全凭证

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

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

申请安全凭证的具体步骤如下：

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

注意：每个账号最多可以拥有两对 SecretId/SecretKey。

2. 生成签名串

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

假设用户的 SecretId 和 SecretKey 分别是：

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

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

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

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

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

2.1. 对参数排序

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

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

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

2.2. 拼接请求字符串

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

然后将格式化后的各个参数用"&"拼接在一起，最终生成的请求字符串为:

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

2.3. 拼接签名原文字符串

此步骤生成签名原文字符串。
签名原文字符串由以下几个参数构成:

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

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

示例的拼接结果为:

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

2.4. 生成签名串

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

具体代码如下，以 PHP 语言为例:

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

```
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));
echo $signStr;
```

最终得到的签名串为：

```
9FzTQAN1UZ489+BqCg1fNBQaCqw=
```

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 9FzTQAN1UZ489+BqCg1fNBQaCqw= ，最终得到的签名串请求参数（Signature）为：9FzTQAN1UZ489%2BBqCg1fNBQaCqw%3D，它将用于生成最终的请求 URL。

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

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

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

4. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理。

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

5. 签名演示

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

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

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

- Signature Demo

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

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

guangzhou&SecretId=AKID********************************&Signature=9FzTQAN1UZ489%2BBqCg1fNBQaCqw%3D&Timestamp=1465185768&Version=2017-03-12 。

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

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

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

Java

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

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

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

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

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

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

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

Python

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

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

import requests

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

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

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

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

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

Golang

```
package main

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

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

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

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

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

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

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

PHP

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

ksort($param);

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

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

Ruby

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

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

```
secret_key = ENV["TENCENTCLOUD_SECRET_KEY"]

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

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

DotNet

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

public class Application {
public static string Sign(string signKey, string secret)
{
string signRet = string.Empty;
using (HMACSHA1 mac = new HMACSHA1(Encoding.UTF8.GetBytes(signKey)))
{
byte[] hash = mac.ComputeHash(Encoding.UTF8.GetBytes(secret));
signRet = Convert.ToBase64String(hash);
}
return signRet;
}
public static string MakeSignPlainText(SortedDictionary<string, string> requestParams, string requestMethod, string reques
tHost, string requestPath)
{
string retStr = "";
retStr += requestMethod;
```

```
retStr += requestHost;
retStr += requestPath;
retStr += "?";
string v = "";
foreach (string key in requestParams.Keys)
{
v += string.Format("{0}={1}&", key, requestParams[key]);
}
retStr += v.TrimEnd('&');
return retStr;
}

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

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

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

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

NodeJS

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

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

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

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

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

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

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

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

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

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

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

返回结果

最近更新时间：2024-12-24 09:13:39

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：2024-12-24 09:13:40

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。

AI 临床助手相关接口
药品适应症接口

最近更新时间：2024-12-24 09:13:37

1. 接口描述

接口请求域名： aca.tencentcloudapi.com 。

药品适应症接口

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：GetDrugIndications。
Version	是	String	公共参数，本接口取值：2021-03-23。
Region	是	String	公共参数，详见产品支持的 地域列表。
Header	是	CommonHeader	请求头
Data	是	GetDrugIndicationsReqData	获取适应症请求

3. 输出参数

参数名称	类型	描述
Code	Integer	错误码 示例值：0
Message	String	错误信息
Data	GetDrugIndicationsResp	适应症响应
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 获取药品适应症示例

获取药品适应症

输入示例

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

{
```

```
"Header": {
"HospitalId": "001",
"Token": "tai.4d563878587a67774d44417a4e544d344e413d3d.1959c3405c614d709babfad5e1b79d54"
},
"Data": {
"Drugs": [
{
"DrugName": "诺氟沙星片",
"Specifications": "0.1g*36片/盒",
"ApprovalNumber": "国药准字H13022772",
"Manufacturer": "石药集团欧意药业有限公司"
}
]
}
}
```

输出示例

```
{
"Response": {
"Code": 0,
"Data": {
"DocInfos": [
{
"DocUrl": "https://dev-aimedical.wecity.qq.com/toolbox/AssistantDetail.html?detailid=04e2bd6dff6422772447fd8c7cd7b5723dd4f7d4&type=drug&token=tai.4d563878587a67774d44417a4e544d344e413d3d.1959c3405c614d709babfad5e1b79d54",
"DrugId": "",
"DrugName": "诺氟沙星片"
}
],
"Indications": [
"感染",
"淋病",
"伤寒",
"其他沙门菌感染",
"肠道感染",
"前列腺炎",
"尿路感染",
"沙门菌感染"
]
},
"Message": "success",
"RequestId": "c6419e0f-7ccb-4fdb-af09-b330d4885f38"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

登录获取辅诊访问Token

最近更新时间：2024-12-24 09:13:37

1. 接口描述

接口请求域名： aca.tencentcloudapi.com 。

登录获取token

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：LoginHisTool。
Version	是	String	公共参数，本接口取值：2021-03-23。
Region	是	String	公共参数，详见产品支持的 地域列表。
Header	是	LoginHeader	请求头 示例值：{ "hospitalId": "1", "platformId": "0", "partnerId": "2" }
Data	是	LoginData	请求体 示例值：{ "doctorId": "123", "doctorName": "医生" }

3. 输出参数

参数名称	类型	描述
Code	Integer	错误码 示例值：0
Message	String	错误信息 示例值：success
Data	LoginDataResp	登录返回token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 辅诊登录

获取访问token

输入示例

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

{
```

```
"Header": {
"HospitalId": "test001",
"PartnerId": "1",
"Timestamp": 1731555285331,
"Signature": "c5146247bc621087037e41952de50ccdcee0a69972cbd8a9cd3631ce79ed91ba",
"PlatformId": "800035384"
},
"Data": {
"DoctorId": "001",
"DoctorName": "abc"
}
}
```

输出示例

```
{
"Response": {
"Code": 0,
"Data": {
"ExpiresIn": 72000,
"Timestamp": 0,
"Token": "tai.4d563878587a67774d44417a4e544d344e413d3d.3ab846e50ec343249fdacba4c79e5ff1"
},
"Message": "success",
"RequestId": "643e3fe3-4ea0-4fcb-b539-f60f7d428bac"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
InternalError	内部错误。
InvalidParameter	参数错误。

登出his工具

最近更新时间：2024-12-24 09:13:37

1. 接口描述

接口请求域名： aca.tencentcloudapi.com 。

登出

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：LoginOutHisTool。
Version	是	String	公共参数，本接口取值：2021-03-23。
Region	是	String	公共参数，详见产品支持的 地域列表。
Header	否	LoginOutHeader	登出请求header
Data	否	LoginOutData	登出请求数据

3. 输出参数

参数名称	类型	描述
Code	Integer	错误码 示例值：0
Message	String	错误信息
Data	LoginOutResponseData	响应数据
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 登出

输入示例

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

{
"Header": {
"PartnerId": "2",
"Timestamp": 1577808000,
"Signature": "721e10a6cd421c8835c70d050831678b21fd48aa9a10e16c744bb10d1976a8ec",
"PlatformId": "123"
},
```

```
"Data": {
"Token": "tai.MTAwMDY%3D.12474ab0-912d-11eb-87e2-4914549178ec"
}
}
```


输出示例


```
{
"Response": {
"Code": 0,
"Message": "success",
"RequestId": "3c140219-cfe9-470e-b241-907877d6fb03"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

智能用药接口

最近更新时间：2024-12-24 09:13:36

1. 接口描述

接口请求域名： aca.tencentcloudapi.com 。

智能用药接口

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：SmartDrugInfo。
Version	是	String	公共参数，本接口取值：2021-03-23。
Region	是	String	公共参数，详见产品支持的 地域列表。
Header	否	CommonHeader	请求头
Data	否	SmartDrugInfoData	药品信息

3. 输出参数

参数名称	类型	描述
Code	Integer	错误码 示例值：0
Message	String	错误信息
Data	SmartDrugInfoResp	智能用药信息
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 智能用药示例

智能用药

输入示例

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

{
"Header": {
"HospitalId": "001",
```

```
"Token": "tai.4d563878587a67774d44417a4e544d344e413d3d.1959c3405c614d709babfad5e1b79d54"
},
"Data": {
"DrugName": "诺氟沙星片",
"Specifications": "0.1g*24片/盒",
"ApprovalNumber": "国药准字H13022772",
"Manufacturer": "石药集团欧意药业有限公司",
"Diagnosis": "中耳炎",
"Age": 65
}
}
```


输出示例


```
{
"Response": {
"Code": 0,
"Data": {
"AdverseReaction": "1. 胃肠道反应 较为常见，可表现为腹部不适或疼痛、腹泻、恶心或呕吐。 2. 中枢神经系统反应 可有头昏、头痛、嗜睡或失眠。 3. 过敏反应 皮疹、皮肤瘙痒，偶可发生渗出性多性红斑及血管神经性水肿。少数患者有光敏反应。 4. 偶可发生： (1)癫痫发作、精神异常、烦躁不安、意识障碍、幻觉、震颤。 (2)血尿、发热、皮疹等间质性肾炎表现。 (3)静脉炎。 (4)结晶尿，多见于高剂量应用时。 (5)关节疼痛。 5. 少数患者可发生血清氨基转移酶升高、血尿素氮增高及周围血象白细胞降低，多属轻度，并呈一过性。",
"ApprovalNumber": "国药准字H13022772",
"Attentions": "1. 本品宜空腹服用，并同时饮水250ml。 2. 由于目前大肠埃希菌对诺氟沙星耐药者多见，应在给药前留取尿标本培养，参考细菌药敏结果调整用药。 3. 本品大剂量应用或尿pH值在7以上时可发生结晶尿。为避免结晶尿的发生，宜多饮水，保持24小时排尿量在1200ml以上。 4. 肾功能减退者，需根据肾功能调整给药剂量。 5. 应用氟喹诺酮类药物可发生中、重度光敏反应。应用本品时应避免过度暴露于阳光，如发生光敏反应需停药。 6. 葡萄糖-6-磷酸脱氢酶缺乏患者服用本品，极个别可能发生溶血反应。 7. 喹诺酮类包括本品可致重症肌无力症状加重，呼吸肌无力而危及生命。重症肌无力患者应用喹诺酮类包括本品应特别谨慎。 8. 肝功能减退时，如属重度(肝硬化腹水)可减少药物清除，血药浓度增高，肝、肾功能均减退者尤为明显，均需权衡利弊后应用，并调整剂量。 9. 原有中枢神经系统疾病患者，例如癫痫及癫痫病史者均应避免应用，有指征时需仔细权衡利弊后应用。请仔细阅读说明书并遵医嘱使用。",
"ChemicalName": "",
"ClinicalResearch": "",
"Contraindication": "对本品及氟喹诺酮类药过敏的患者禁用。",
"DocRevisionTime": "",
"DrugBasicCode": "86902770001457",
"DrugDosageForm": "片剂",
"DrugHashId": "3f8ce4e9cbc3ab786bb48f8f9ea252df1eec9e92",
"DrugId": "",
"DrugName": "诺氟沙星片",
"DrugRoute": "口服给药",
"ElderlyPatients": "老年患者常有肾功能减退，因本品部分经肾排出，需减量应用。",
"EnglishChemicalName": "",
"EnglishName": "Norfloxacin Tablets",
"EnglishTradeName": "",
"ExpireDate": "24 月",
"ImgUrl": "https://p.qpic.cn/wyp_pic/Q3auHgzwzM6n8rbxOAOExJI4ch8eCtfG61UEvbvPSQLdHoMs8I9V28uHdDBujXsCOe1lwDjm9aIWHia198RMb3zQzviaCt5Und/0",
"Indications": "适用于敏感细菌所引起的急、慢性肾盂肾炎、膀胱炎、前列腺炎、细菌性痢疾、胆囊炎、伤寒、产前产后感染、盆腔炎、中耳炎、鼻窦炎、急性扁桃腺炎及皮肤软组织感染等，也可作为腹腔手术的预防用药。",
"Ingredients": "本品主要成份为：诺氟沙星。",
"Interactions": "1. 尿碱化剂可减低本品在尿中的溶解度，导致结晶尿和肾毒性。 2. 本品与茶碱类合用时可能由于与细胞色素P450结合部位的竞争性抑制，导致茶碱类的肝清除明显减少，血消除半衰期(t1/2β)延长，血药浓度升高，出现茶碱中毒症状，如恶心、呕吐、震颤、不安、激动、抽搐、心悸等，故合用时应测定茶碱类血药浓度和调整剂量。 3. 环孢素与本品合用，可使前者的血药浓度升高，必须监测环孢素血药浓度，并调整剂量。 4. 本品与抗凝药华法林同用时可增强后者的抗凝作用，合用时应严密监测患者的凝血酶原时间。 5. 丙磺舒可减少本品自肾小管分泌约50%，合用时可因本品血浓度增高而产生毒性。 6. 本品与呋喃妥因具拮抗作用，不推荐联合应用。 7. 多种维生素，或其他含铁、锌离子的制剂及含铝或镁的制酸药可减少本品的吸收，建议避免合用，不能避免时在本品服药前2小时，或服药后6小时服用。 8. 去羟肌苷(didanosine，DDI)可减少本品的口服吸收，因其制剂中含铝及镁，可与氟喹诺酮类螯合，故不宜合用。 9. 本品干扰咖啡因的代谢，从而导致咖啡因清除减少，血消除半衰期(t1/2β)延长，并可能产生中枢神经系统毒性。",
```

```
"Manufacturer": "石药集团欧意药业有限公司",
"OctTag": "",
"OtherNames": "",
"Overdose": "尚不明确。",
"Pack": "0.1g*36片",
"PediatricDrugs": "18岁以下的患者禁用。",
"Pharmacokinetics": "健康成人空腹一次口服本品0.4克，1～2小时血药峰浓度(Cmax)为1.32mg/ml，8小时后尚有0.33μg/ml，吸收半衰期为0.41小时，分布相半衰期(t1/2α)为0.41小时，消除相半衰期(t1/2β)为2.54小时。12小时内尿中平均排出量占给药量的27.88%，其中22.56%在6小时内排出。 大鼠口服本品后，迅速分布各组织间，肾、肝、淋巴结、脾、颚下腺、血浆等的药物浓度均较高，上颚扁桃体及兔的皮肤也有分布。",
"PharmacologyToxicology": "本品为氟喹诺酮类抗菌药，具广谱抗菌作用，尤其对需氧革兰阴性杆菌的抗菌活性高，对下列细菌在体外具良好的抗菌作用：肠杆菌科的大部分细菌，包括枸橼酸杆菌属、阴沟肠杆菌、产气肠杆菌等肠杆菌属、大肠埃希菌、克雷伯菌属、变形菌属、沙门菌属、志贺菌属、克雷伯氏菌属、变形杆菌属、产气肠杆菌、沙雷氏菌属、枸橼酸菌属等对本品高度敏感，最低抑菌浓度一般低于0.5μg/ml。流感杆菌对本品亦高度敏感。对铜绿假单胞菌及其它假单孢菌亦有作用，但需要较高的浓度，最低抑菌浓度在4～32μg/ml；由于在尿中浓度超过上述浓度数倍至十多倍，用于治疗铜绿假单胞菌泌尿系统感染仍然有效。在革兰阳性球菌中，对金黄色葡萄球菌及表皮葡萄球菌较敏感，抑制90%细菌的最低浓度约为1～2μg/ml，对肠球菌和肺炎球菌的抑制90%细菌的最低浓度约为4μg/ml和16μg/ml。",
"PhenotypicTrait": "本品为糖衣片。除去糖衣后显淡黄色。对光、热、湿较稳定。",
"Pinyin": "NuoFuShaXingPian",
"PregnantAndLactatingWomen": "曾用猴进行繁殖研究，剂量高达人用量的10倍，发现本品可致流产。该剂量在猴的血浆峰浓度(Cmax)约为人的2倍。本品在动物中并未证实有致畸作用。然而，在孕妇并未进行合适的、有良好对照的研究，因此本品不宜用于孕妇。 本品是否经乳汁分泌尚缺乏资料。当乳妇应用200mg本品时，乳汁中不能检出该药。然而，由于研究剂量较小，且本类药物的其他品种经乳汁分泌，加之对新生儿及婴幼儿潜在的严重不良反应，乳妇应避免应用本品或于应用时停止哺乳。",
"Property": "",
"RecommendedUsage": {
"Frequency": "",
"SingleDose": "",
"UsageRoute": ""
},
"References": "",
"SequenceId": 27668,
"Specifications": "0.1g*24片/盒",
"Storage": "密封，干燥处保存。",
"TradeName": "石药/石药集团",
"UsageAndDosage": "1．大肠埃希菌、肺炎克雷伯菌及奇异变形菌所致的急性单纯性下尿路感染 一次400mg，一日2次，疗程3日。 2．其他病原菌所致的单纯性尿路感染 剂量同上，疗程7～10日。 3．复杂性尿路感染 剂量同上，疗程10～21日。 4．单纯性淋球菌性尿道炎 单次800～1200mg。 5．急性及慢性前列腺炎 一次400mg，一日2次，疗程28日。 6．肠道感染 一次300～400mg，一日2次，疗程5～7日。 7．伤寒沙门菌感染 一日800～1200mg，分2～3次服用，疗程14～21日。",
"Warning": ""
},
"Message": "success",
"RequestId": "f77dbf91-2d35-4047-bccb-dc4dde4c06b7"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

智能预测

最近更新时间：2024-12-24 09:13:36

1. 接口描述

接口请求域名： aca.tencentcloudapi.com 。

辅诊智能预测接口

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：SmartPredict。
Version	是	String	公共参数，本接口取值：2021-03-23。
Region	是	String	公共参数，详见产品支持的 地域列表。
Header	是	CommonHeader	请求头
Data	是	SmartPredictReqData	请求体

3. 输出参数

参数名称	类型	描述
Code	Integer	错误码 示例值：0
Message	String	错误信息 示例值：success
Data	SmartPredictRespData	智能预测内容
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 智能预测

合理用药+辅助决策

输入示例

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

{
"Header": {
"Token": "tai.MV8wXzgwMDAwMDUyOA%3D%3D.NQ%3D%3D.bb6f0c70-ff5a-11ed-8d21-2fc84f9d28c2",
```

```
"HospitalId": "1"
},
"Data": {
"RequestCase": {
"CaseType": 0,
"PatientBaseinfo": {
"Name": "陈",
"Sex": "female",
"Age": "22y",
"PatientId": "123",
"BirthDay": "2000-01-01 00:00:00",
"Height": "165",
"Weight": "55"
},
"ChiefComplaint": "患者怀孕20周+3天。自诉头晕月余，左前胸（乳头上方区域）疼痛伴呼吸短促3天，凌晨加重，自服硝酸甘油片未缓解，今晨因剧烈咳嗽伴血性咳出物入院。",
"PresentIllness": "胸痛5年；支气管扩张2年；一月前出现过一过性高血糖昏迷。",
"PhysicalExam": {
"Weight": "65",
"Pulse": "70",
"Breathe": "20"
},
"EmrDiagnosises": [
{
"IcdCode": "J06.900",
"DiagnosisName": "急性上呼吸道感染"
},
{
"IcdCode": "A49.003",
"DiagnosisName": "耐甲氧西林金黄色葡萄球菌感染"
},
{
"IcdCode": "I21.001",
"DiagnosisName": "急性ST段抬高型前壁心肌梗塞"
}
],
"Department": "测试科室",
"CaseId": "a123",
"CaseTime": "2022-08-11 16:57:28",
"DoctorInfo": {
"DoctorId": "123",
"DoctorName": "张三"
},
"VisitId": "1234",
"Prescriptions": [
{
"DrugId": "01008",
"DrugName": "阿莫西林分散片",
"DosagePerTime": "100",
"DosagePerTimeUnit": "片",
"TimePerDay": "每周一次",
"Usage": "静脉注射-注射",
"PrescriptionId": "752d07c9-af98-48be-870c-e798f4d78f92",
"BeginTime": 1660278229,
"EndTime": 1660278229
}
]
```

```
},
"RequestType": 0
}
}
```

输出示例


```
{
"Response": {
"Code": 0,
"Message": "success",
"RequestId": "6354b58e-ff37-40c4-84a4-aa71e80a1608",
"Data": {
"RationalDrugInfo": {
"Hit": true,
"Abnormals": [
{
"Type": "PrescriptionError",
"Title": "处方信息缺失",
"AbnormalReason": "患儿体重缺失"
}
],
"DrugRoutes": [
{
"DrugId": "10003637",
"DrugName": "依折麦布片1",
"RiskTips": "依折麦布片1给药途径为：静脉注射2，但说明书给药途径为：胃肠道给药-口腔给药-口服，存在给药途径风险",
"RiskLevel": "中级风险"
}
],
"DrugIndications": [
{
"DrugId": "10003637",
"DrugName": "依折麦布片1",
"RiskTips": "依折麦布片1的适应症为：冠心病,脑血管病,高脂血症,纯高胆固醇血症,原发性高胆固醇血症,冠状动脉粥样硬化...，但患者诊断为：2型糖尿病2,
咳嗽，存在适应症风险",
"RiskLevel": "低级风险"
}
],
"DrugList": [
{
"DrugId": "10003637",
"DrugName": "依折麦布片1",
"DocUrl": "https://dev-aimedical.wecity.qq.com/toolbox/AssistantDetail.html?detailid=3ba19002112265c9849ca41b9508bcc256052
a59&drugOrgId=test001@10003637&type=drug&token=tai.4d563878587a67774d44417a4e544d344e413d3d.41f05483dca54f5689f1b75769b3b8
b2",
"NotFound": false
}
]
}
}
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

科室同步接口

最近更新时间：2024-12-24 09:13:35

1. 接口描述

接口请求域名： aca.tencentcloudapi.com 。

用于院方科室管理，获取科室列表和状态、新增或修改科室信息、删除科室。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：SyncDepartment。
Version	是	String	公共参数，本接口取值：2021-03-23。
Region	是	String	公共参数，详见产品支持的 地域列表。
Header	是	CommonHeader	请求头
Data	是	SyncDepartmentData	同步数据

3. 输出参数

参数名称	类型	描述
Code	Integer	错误码 示例值：0
Message	String	错误信息
Data	SyncDepartmentRespData	响应数据
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId ）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 同步科室

新增科室数据

输入示例

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

{
"Header": {
"HospitalId": "001",
```

```
"Token": "tai.4d563878587a67774d44417a4e544d344e413d3d.1959c3405c614d709babfad5e1b79d54"
},
"Data": {
"Cmd": 2,
"List": [
{
"Id": "123",
"Name": "综合门诊",
"Scope": 2,
"OutpatientOn": true,
"InHospitalOn": true
}
]
}
}
```


输出示例


```
{
"Response": {
"Code": 0,
"Data": {
"List": []
},
"Message": "success",
"RequestId": "a2c89afa-1b0d-40dc-87f3-acd0583c21bf"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

同步标准字典接口

最近更新时间：2024-12-24 09:13:35

1. 接口描述

接口请求域名： aca.tencentcloudapi.com 。

同步标准字典，如给药频次、给药途径、科室、诊断等

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：SyncStandardDict。
Version	是	String	公共参数，本接口取值：2021-03-23。
Region	是	String	公共参数，详见产品支持的 地域列表。
Header	是	CommonHeader	请求头
Data	是	SyncDictData	字典数据

3. 输出参数

参数名称	类型	描述
Code	Integer	错误码 示例值：0
Message	String	错误信息
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 同步给药频次

同步给药频次

输入示例

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

{
"Header": {
"HospitalId": "001",
"Token": "tai.4d563878587a67774d44417a4e544d344e413d3d.1959c3405c614d709babfad5e1b79d54"
},
```

```
"Data": {
"HospitalId": "001",
"DictType": 1,
"Dicts": [
{
"FreqCode": "qid",
"FreqName": "一日三次",
"Disable": 0
}
]
}
}
```


输出示例


```
{
"Response": {
"Code": 0,
"Message": "success",
"RequestId": "77f9c09c-8db0-40e9-8293-d4620e82cc9b"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

药品同步

最近更新时间：2024-12-24 09:13:35

1. 接口描述

接口请求域名： aca.tencentcloudapi.com 。

药品同步，一次同步数据不要超过500个

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：UploadDrugs。
Version	是	String	公共参数，本接口取值：2021-03-23。
Region	是	String	公共参数，详见产品支持的 地域列表。
Header	否	CommonHeader	请求头数据
Data	否	UploadDrugData	药品上传数据

3. 输出参数

参数名称	类型	描述
Code	Integer	错误码 示例值：0
Message	String	错误信息
Data	OperateResp	操作信息
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 药品同步

输入示例

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

{
"Header": {
"HospitalId": "1",
"Token": "abc"
```

```
},
"Data": {
"Drugs": [
{
"DrugOrgId": "医院药品id",
"DrugName": "医院药品通用名",
"DrugCommodityName": "医院药品商品名",
"Specifications": "医院药品规格",
"ApprovalNumber": "医院药品批准文号",
"Manufacturer": "医院厂商",
"DosageForm": "剂型",
"DosageFormCode": "剂型编码",
"DefinedDailyDose": "抗菌药DDD值",
"Amount": "药品单价",
"YbCode": "国家医保编码",
"DrugBasicCode": "药品本位码",
"Unuse": 0,
"PropertyInfo": {
"DrugType": 1,
"AntibacterialType": 0,
"AntibacterialClass": 0,
"SpeciallyDrugType": 0,
"IsBasicDrug": 1,
"ChargeType": 2
}
}
]
}
}
```

输出示例

```
{
"Response": {
"Code": 0,
"Message": "成功",
"RequestId": "3c140219-cfe9-470e-b241-907877d6fb03",
"Data": {
"Dummy": false
}
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

数据结构

最近更新时间：2024-12-24 09:13:38

Abnormals

异常提醒

被如下接口引用：SmartPredict。

名称	类型	必选	描述
Type	String	是	类型 示例值：PrescriptionError
Title	String	是	标题 示例值：人群特征不符
AbnormalReason	String	是	异常原因 PatientInfo 病人信息缺失；OrderInfo 医嘱信息缺失； PrescriptionError 处方异常提醒 示例值：男性人群，与阴道炎诊断冲突

CommonHeader

业务接口请求头

被如下接口引用：GetDrugIndications, SmartDrugInfo, SmartPredict, SyncDepartment, SyncStandardDict, UploadDrugs。

名称	类型	必选	描述
HospitalId	String	是	机构ID 示例值：1
Token	String	是	合作方ID 示例值：1

CriticalInfo

危疾重症

被如下接口引用：SmartPredict。

名称	类型	必选	描述
Type	Integer	是	危急重症类型 0:文字描述类 1:数值检查类 示例值：0
Tips	String	是	提示 示例值：该患者存在以下危急重症：呼吸困难，考虑可能，请注意

CurrentDiagnosis

当前诊断

被如下接口引用：SmartPredict。

名称	类型	必选	描述
DiagnoseDisease	String	是	诊断疾病 示例值：急性上呼吸道感染
DiseaseGuideInfo	String	是	疾病指南信息 示例值：http://127.0.0.1:8080/toolbox/AssistantDetail.html?detailid=1497818&type=disease&token=tai.MV8wXzgwMDAwMDUyOA%3D%3D.NQ%3D%3D.bb6f0c7ff5a-11ed-8d21-2fc84f9d28c2
StandardName	String	是	标准名称 示例值：急性上呼吸道感染

Department

科室信息

被如下接口引用：SyncDepartment。

名称	类型	必选	描述
Id	String	是	科室ID 注意：此字段可能返回 null，表示取不到有效值。 示例值：123
Name	String	是	科室名称 注意：此字段可能返回 null，表示取不到有效值。 示例值：综合门诊
Scope	Integer	是	科室类型 0:门诊 1:住院 2:门诊+住院 注意：此字段可能返回 null，表示取不到有效值。 示例值：2
OutpatientOn	Boolean	是	门诊区开关 true:此科室对应的门诊区开启智能审方功能, false:此科室对应的门诊区关闭智能审方功能; 仅对scope为0/2的科室生效 注意：此字段可能返回 null，表示取不到有效值。 示例值：True
InHospitalOn	Boolean	是	住院区开关 true:此科室对应的住院区开启智能审方功能, false:此科室对应的住院区关闭智能审方功能; 仅对scope为1/2的科室生效 注意：此字段可能返回 null，表示取不到有效值。 示例值：True

DiagnosisInfo

诊断、辅助内容

被如下接口引用：SmartPredict。

名称	类型	必选	描述
IntentType	Integer	否	默认0，0:初诊-常规诊疗 1:复诊 2:检验检查/取药/咨询/疫苗 3:信息缺失 4:信息错误 示例值：0
RiskInfo	String	否	诊断风险
SuspectedDiagnosis	Array of SuspectedDiagnosis	否	疑似诊断列表
ReferralInfo	ReferralInfo	否	转诊提醒
CriticalInfo	Array of CriticalInfo	否	危急重症
VitalSignsInfo	VitalSignsInfo	否	生命体征风险
DifferDiagnosis	Array of DifferDiagnosis	否	鉴别诊断
RecordQuality	RecordQuality	否	病历质控
CurrentDiagnosis	Array of CurrentDiagnosis	否	当前诊断
TreatmentGuide	Array of TreatmentGuide	否	治疗方案
EmrQuality	EmrQuality	否	病历质控
HealthPrescriptions	Array of HealthPrescriptions	否	健康处方

Dict

字典信息

被如下接口引用：SyncStandardDict。

名称	类型	必选	描述
FreqCode	String	否	给药频次编码 示例值：qid
FreqName	String	否	给药频次名称 示例值：一日三次
Disable	Integer	否	是否禁用 0-启用 1-禁用 示例值：false
UsageCode	String	否	给药途径编码 示例值：iv
UsageName	String	否	给药途径名称 示例值：肌肉注射
DeptId	String	否	科室ID 示例值：a01
DeptName	String	否	科室名称 示例值：内科
Scope	Integer	否	科室区域类型 0:门诊 1:住院 2:门诊+住院 示例值：1
OutpatientOn	Boolean	否	门诊开关 示例值：True
InHospitalOn	Boolean	否	住院 示例值：false
DiagCode	String	否	诊断编码 示例值：4567890
DiagName	String	否	诊断名称 示例值：痛经
IcdCode	String	否	ICD代码 示例值：N94.600

DifferDiagnosis

鉴别诊断

被如下接口引用：SmartPredict。

名称	类型	必选	描述
DifferName	String	是	鉴别名称 示例值：过敏性鼻炎
DifferTips	String	是	鉴别提示 示例值：从症状鉴别：1、起病急骤、鼻腔发痒、喷嚏频繁、鼻涕呈清水样，无发热，咳嗽较少；2、多由过敏因素如螨虫、灰尘、动物皮毛、低温等刺激引起；3、如脱离过敏源，数分钟及1～2小时内症状即消失；从检查鉴别：1、体检可见鼻黏膜苍白水肿；2、鼻分泌物涂片可见嗜酸性粒细胞增多
DiseaseGuideInfo	String	是	疾病指南信息 示例值：http://127.0.0.1:8080/toolbox/AssistantDetail.html?detailid=1498034&type=disease&token=tai.MV8wXzgwMDAwMDUyOA%3D%3D.NQ%3D%3D.bb6f0cff5a-11ed-8d21-2fc84f9d28c2

DocInfo

药品文档信息

被如下接口引用：GetDrugIndications。

名称	类型	描述
DrugId	String	药品ID 示例值：123
DrugName	String	药品名称 示例值：诺氟沙星片
DocUrl	String	说明书地址 示例值：http://ip:port/toolbox/AssistantDetail.html?detailid=04e2bd6dff6422772447fd8c7cd7b5723dd4f7d4&type=drug&token=tai.MV8wXzEwMDA2.NQ==.54724330-1878-11ed-b1d0-b72c69c6d6f1

DoctorInfo

医生信息

被如下接口引用：SmartPredict。

名称	类型	必选	描述
DoctorId	String	否	医生ID
DoctorName	String	否	医生姓名
DoctorPhone	String	否	医生电话

Drug

药品信息

被如下接口引用：UploadDrugs。

名称	类型	必选	描述
DrugOrgId	String	是	医院药品id
DrugName	String	是	医院药品通用名
DrugCommodityName	String	是	医院药品商品名
Specifications	String	是	医院药品规格
ApprovalNumber	String	是	医院药品批准文号
Manufacturer	String	是	生产厂商
DosageForm	String	是	剂型
Unuse	Integer	是	使用状态 0:启用 1:停用 示例值：0
DosageFormCode	String	否	剂型编码
DefinedDailyDose	String	否	抗菌药DDD值
Amount	String	否	药品单价 示例值：10
YbCode	String	否	国家医保编码
DrugBasicCode	String	否	药品本位码
PropertyInfo	DurgPropertyInfo	否	药品属性

DrugList

药品列表

被如下接口引用：SmartPredict。

名称	类型	必选	描述
DrugId	String	否	药品ID 示例值：001
DrugName	String	否	药品名称
DocUrl	String	否	文档地址
NotFound	Boolean	否	是否找到 示例值：false

DrugUsage

处方药品信息

被如下接口引用：SmartPredict。

名称	类型	必选	描述
DrugId	String	是	药品ID 示例值：01008
DrugName	String	是	药品名称 示例值：阿莫西林分散片
TimePerDay	String	是	日服药频次 示例值：q4h
Usage	String	是	给药途径 示例值：静脉注射-注射
PrescriptionId	String	是	处方ID，药品不同分组是传不同的处方ID 示例值：752d07c9-af98-48be-870c-e798f4d78f92
DosagePerTime	String	否	每次剂量 示例值：100
DosagePerTimeUnit	String	否	每次剂量单位 示例值：片
Time	String	否	单次服药时间 示例值：餐前一小时
Cycle	String	否	给药周期 示例值：每隔四小时
DosagePerDay	String	否	单日剂量 示例值：1
Course	String	否	疗程 示例值：2周一疗程
Speed	String	否	给药速度 示例值：5g本品于5～10分钟内滴完
BeginTime	Integer	否	处方生效时间戳，住院医嘱必须传(caseType =1) 示例值：1660278229
EndTime	Integer	否	处方失效时间戳，住院医嘱必须传(caseType =1) 示例值：1660278229
Package	String	否	开具数量 示例值：1
PackageUnit	String	否	开具数量单位 示例值：粒
GroupInj	String	否	相同标志液体间进行配伍禁忌审核，不同标志间液体不进行配伍禁忌审核 示例值：1

名称	类型	必选	描述
PrescriptionCharge	String	否	处方金额 示例值：10
MedicationDays	String	否	用药天数 示例值：10

DurgPropertyInfo

药品属性

被如下接口引用：UploadDrugs。

名称	类型	必选	描述
DrugType	Integer	否	药品类型 1:西药,2:中成药,3:中药,4:化学药品,5:生物制药 示例值：1
AntibacterialType	Integer	否	抗菌药分类 1:抗真菌药物, 2:抗细菌药物, 3:抗结核药物, 4:其他抗菌药, 0:普通药品 示例值：1
AntibacterialClass	Integer	否	抗菌药级别 1:非限制级, 2:限制级, 3:特殊使用级 示例值：1
SpeciallyDrugType	Integer	否	特殊药品类型 1:毒性药品, 2:麻醉药品, 3:放射药品, 4:精神一类药品, 5:精神二类药品, 6:其他特管药品， 7:贵重药品 示例值：1
IsBasicDrug	Integer	否	是否为基本药物 1:是, 2:否, 0:未知 示例值：1
ChargeType	Integer	否	社保药品 1:甲类药品, 2:乙类药品, 3:双跨药品, 4:自费药品, 0:未知 示例值：2

EmrDiagnosises

诊断列表

被如下接口引用：SmartPredict。

名称	类型	必选	描述
DiagnosisName	String	是	诊断名称 示例值：急性上呼吸道感染
IcdCode	String	否	ICD代码 示例值：J06.900

EmrQuality

病历质控

被如下接口引用：SmartPredict。

名称	类型	必选	描述
MissPhysicalExamination	Array of String	否	缺失体格检查项目

GetDrugIndicationsReqData

获取适应症请求

被如下接口引用：GetDrugIndications。

名称	类型	必选	描述
Drugs	Array of IndicationsDrug	否	查询药品列表

GetDrugIndicationsResp

药品适应症响应

被如下接口引用：GetDrugIndications。

名称	类型	描述
Indications	Array of String	适应症
DocInfos	Array of DocInfo	药品说明

HealthPrescriptions

健康处方

被如下接口引用：SmartPredict。

名称	类型	必选	描述
Title	String	否	标题
Url	String	否	健康处方链接
KeyInformation	String	否	关键信息

IndicationsDrug

适应症药品请求

被如下接口引用：GetDrugIndications。

名称	类型	必选	描述
DrugName	String	是	药品名称 示例值：诺氟沙星片
Specifications	String	是	规格 示例值：0.1g*36片/盒
ApprovalNumber	String	是	批准文号 示例值：国药准字H13022772
Manufacturer	String	是	生产厂家 示例值：石药集团欧意药业有限公司
DrugId	String	否	药品ID 示例值：123
TradeName	String	否	商品名
Type	Integer	否	类型 默认0 0-西药 2-中药 示例值：0

LoginData

登录请求对象

被如下接口引用：LoginHisTool。

名称	类型	必选	描述
DoctorId	String	是	医生ID 示例值：001
DoctorName	String	否	医生名称 示例值：test
DoctorLevel	String	否	医生职级 主治医师、副主任医师、主任医师

名称	类型	必选	描述
			示例值：主治医师
DoctorDepartment	String	否	医生科室 当前登录科室
DepartmentId	String	否	科室ID

LoginDataResp

登录返回数据

被如下接口引用：LoginHisTool。

名称	类型	必选	描述
Token	String	否	token
ExpiresIn	Integer	否	过期时间 示例值：7200
Timestamp	Integer	否	服务端时间戳，时间戳校验失败时返回 示例值：1730791776988

LoginHeader

登录接口请求头

被如下接口引用：LoginHisTool。

名称	类型	必选	描述
HospitalId	String	是	机构ID 示例值：1
PartnerId	String	是	合作方ID 示例值：1
Timestamp	Integer	是	加密时间戳毫秒 示例值：1730791776988
Signature	String	是	签名数据
PlatformId	String	否	平台ID，平台版登录时传入 示例值：001

LoginOutData

登出

被如下接口引用：LoginOutHisTool。

名称	类型	必选	描述
Token	String	是	登录获取的token

LoginOutHeader

登出header对象

被如下接口引用：LoginOutHisTool。

名称	类型	必选	描述
PartnerId	String	是	合作方ID 示例值：2
Timestamp	Integer	是	时间戳毫秒数

名称	类型	必选	描述
Signature	String	是	签名值
HospitalId	String	否	医院ID 单院版传这个
PlatformId	String	否	平台ID 平台版传这个

LoginOutResponseData

登出数据

被如下接口引用：LoginOutHisTool。

名称	类型	描述
Timestamp	Integer	服务器时间戳毫秒

OperateResp

操作结果

被如下接口引用：UploadDrugs。

名称	类型	描述
Dummy	Boolean	操作结果 示例值：true

PatientBaseInfo

患者信息

被如下接口引用：SmartPredict。

名称	类型	必选	描述
Sex	String	是	性别 示例值：female
Height	String	是	身高 单位cm 示例值：170
Weight	String	是	体重 单位kg 示例值：110
PatientId	String	是	患者ID
Name	String	否	名称 示例值：姓名
Age	String	否	年龄 示例值：22y
BirthPlace	String	否	出生地
LivePlace	String	否	居住地
BirthDay	String	否	出生日期和年龄必须传一个 示例值：2000-01-01 00:00:00

PatientFamilyHistory

患者家族病史

被如下接口引用：SmartPredict。

名称	类型	必选	描述
FamilyDiseaseHistory	String	否	家族病史 如家族病史不能分开，可传入此字段
Relation	String	否	关系
CurrentSituation	String	否	当前情况

PatientHistory

患者过往病史

被如下接口引用：SmartPredict。

名称	类型	必选	描述
DiseaseHistory	String	否	病史
TreatmentHistory	String	否	治疗史

PhysicalExam

体格检查

被如下接口引用：SmartPredict。

名称	类型	必选	描述
Pulse	String	是	脉搏，单位：次/分 示例值：70
Breathe	String	是	呼吸，单位：次/分 示例值：20
Weight	String	否	体重，单位KG
BodyTemperature	String	否	体温，单位：℃
DiastolicPressure	String	否	舒张压，单位：mmHg
SystolicPressure	String	否	收缩压，单位：mmHg

RationalDrugInfo

合理用药信息

被如下接口引用：SmartPredict。

名称	类型	必选	描述
Hit	Boolean	否	是否有风险 示例值：true
DrugUsages	Array of RiskInfo	否	药品用量风险
DrugRepeats	Array of RiskInfo	否	重复用药风险
DrugRoutes	Array of RiskInfo	否	用药途径风险
SpecialPopulations	Array of RiskInfo	否	特殊人群风险
DrugTaboos	Array of RiskInfo	否	禁忌症风险
DrugInteractions	Array of RiskInfo	否	相互作用风险
DrugIncompatibility	Array of RiskInfo	否	配伍禁忌风险
DrugAllergys	Array of RiskInfo	否	过敏风险
DrugIndications	Array of RiskInfo	否	适应症风险

名称	类型	必选	描述
Abnormals	Array of Abnormals	否	异常提醒
DrugList	Array of DrugList	否	药品列表

RecommendedUsage

推荐用法

被如下接口引用：SmartDrugInfo。

名称	类型	描述
UsageRoute	String	给药途径 示例值：口服
Frequency	String	给药频率 格式为“最小频次,最大频次,频次单位,频次周期”，如"1,2,次,2"， 表示2天内最少给药1次，最大给药2次。 示例值：2.0,2.0,次,1.0
SingleDose	String	给药剂量 格式为"最小剂量,最大剂量,剂量单位"，如"10,10,ml"， 表示每次最大给药量为10ml，最小给药量为10ml。 示例值：400,400,mg

RecordQuality

病历质控

被如下接口引用：SmartPredict。

名称	类型	必选	描述
Hit	Boolean	是	病历是否有问题 示例值：True
Completeness	String	否	完整性问题
Timeliness	String	否	及时性问题
Logical	String	否	逻辑性问题 示例值：病历中怀孕术语书写不规范，建议规范书写为：妊娠；病历中急性ST段抬高型前壁心肌梗塞术语书写不规范，建议规范书写为：

ReferralInfo

转诊提醒

被如下接口引用：SmartPredict。

名称	类型	必选	描述
Hit	Boolean	是	命中 示例值：True
Tips	String	是	提示 示例值：该患者患有急性上呼吸道感染，超出诊疗范围，建议转诊

RequestCase

预测数据

被如下接口引用：SmartPredict。

名称	类型	必选	描述
CaseType	Integer	是	处方类型 0:门诊处方；1:住院医嘱；2:急诊处方 示例值：0

名称	类型	必选	描述
ChiefComplaint	String	是	主诉 示例值：头疼一天
Department	String	是	科室 示例值：口腔科
CaseId	String	是	病历文书ID 医生每次书写病历文书的ID，文书内容包含主诉，病史，当前诊断等内容 门诊场景：门诊病历文书（带有主诉、病史、诊断及药品的）只有一份，这个编号只有一个。 住院场景：假设住院3天，医生每天都会写一份病历文书（带有主诉、病史、诊断及医嘱药品的文书），那么有对应三个病历文书编号，每次调用预测接口都要传入不同的病历文书编号。注意：如两次调用预测接口，传相同的caseid，则在药师端管理平台的上一次审方记录中的诊断会被本次接口传入的诊断更新。 示例值：a123
CaseTime	String	是	病历更新时间 示例值：2022-08-11 16:57:28
VisitId	String	是	就诊ID 门诊处方传门诊号，住院医嘱传住院号；备注：门诊场景：用户挂一次号，看一个医生，这时候会有一个代表变成就诊的编号，下一次挂号就诊，这个编号会变。 住院场景：用户本次办理入院，会有一个住院编号，仅代表本次住院，如果下次再住院，这个编号会变。 示例值：1234
PatientBaseinfo	PatientBaseInfo	是	患者信息
DoctorInfo	DoctorInfo	是	医生信息
PresentIllness	String	否	现病史 示例值：胸痛5年；支气管扩张2年；一月前出现过一过性高血糖昏迷。
PatientOther	String	否	患者其他信息，包含过敏史等
PatientHistory	PatientHistory	否	患者过往病史
PatientFamilyHistory	PatientFamilyHistory	否	患者家族病史
PhysicalExam	PhysicalExam	否	体格检查
EmrDiagnosises	Array of EmrDiagnosises	否	诊断列表，第一个为首要诊断
Prescriptions	Array of DrugUsage	否	处方列表

RiskInfo

风险信息

被如下接口引用：SmartPredict。

名称	类型	必选	描述
DrugId	String	否	药品ID
DrugName	String	否	药品名称
RiskLevel	String	否	风险等级：低级风险、中级风险、高级风险 示例值：低级风险
RiskTips	String	否	风险提示
FdaLevel	String	否	FDA分级
RelatedDrugName	String	否	关联药品名称
RelatedPrescriptionId	String	否	关联处方ID

SmartDrugInfoData

智能用药请求数据

被如下接口引用：SmartDrugInfo。

名称	类型	必选	描述
DrugName	String	是	药品名称 示例值：诺氟沙星片
Specifications	String	是	规格 示例值：0.1g*36片/盒
ApprovalNumber	String	是	批准文号 示例值：国药准字H13022772
Manufacturer	String	是	生产厂家 示例值：石药集团欧意药业有限公司
DrugId	String	否	药品ID 示例值：123
Diagnosis	String	否	诊断 示例值：尿路感染
Age	Float	否	年龄 示例值：20

SmartDrugInfoResp

智能用药响应

被如下接口引用：SmartDrugInfo。

名称	类型	描述
DrugId	String	药品ID 示例值：123
SequenceId	Integer	序列ID 示例值：10031837
DrugHashId	String	药品哈希ID 示例值：3f8ce4e9cbc3ab786bb48f8f9ea252df1eec9e92
ImgUrl	String	图片URL 示例值： https://p.qpic.cn/wyp_pic/Q3auHgzwzM6n8rbxOAOExJl4ch8eCtfG61UEvbvPSQI
DrugName	String	药品名称 示例值：诺氟沙星片
TradeName	String	商品名
EnglishName	String	英文名称 示例值：Norfloxacin Tablets
EnglishTradeName	String	英文商品名
Pinyin	String	拼音 示例值：NuoFuShaXingPian
OtherNames	String	其他名称
ChemicalName	String	化学名称
EnglishChemicalName	String	英文化学名称
ApprovalNumber	String	批准文号 示例值：国药准字H13022772

名称	类型	描述
Property	String	药品属性标签 多个标签时 | 分隔，如抗菌药|抗生素|贵重药品
Ingredients	String	药品成分
PhenotypicTrait	String	药品性状 示例值：本品为糖衣片。除去糖衣后显淡黄色。对光、热、湿较稳定。
Indications	String	适应症 示例值：适用于敏感细菌所引起的急、慢性肾盂肾炎、膀胱炎、前列腺炎、细菌性痢疾、胆囊炎、伤 可作为腹腔手术的预防用药。
Specifications	String	规格 示例值：0.1g*24片/盒
UsageAndDosage	String	用法用量 示例值：空腹口服：成人一次0.1～0.2g，一日3～4次；重症酌情加量，一日1.6g，分4次服用。
RecommendedUsage	RecommendedUsage	推荐用法
AdverseReaction	String	不良反应 示例值：可偶有头晕、头痛及消化道反应等症状，个别病人可有血清谷丙氨基转移酶及尿素氮升高，
Contraindication	String	禁忌 示例值：对本品及氟喹诺酮类药过敏的患者禁用。
Attentions	String	注意事项 示例值：有过敏史者及严重肾功能不全者慎用石药诺氟沙星片。
Overdose	String	药物过量 示例值：尚不明确。
PregnantAndLactatingWomen	String	孕妇及哺乳期妇女用药 示例值：曾用猴进行繁殖研究，剂量高达人用量的10倍，发现本品可致流产。该剂量在猴的血浆峰浓 合适的、有良好对照的研究，因此本品不宜用于孕妇。本品是否经乳汁分泌尚缺乏资料。当乳妇应用 品种经乳汁分泌，加之对新生儿及婴幼儿潜在的严重不良反应，乳妇应避免应用本品或于应用时停止
ElderlyPatients	String	老年患者用药 示例值：老年患者常有肾功能减退，因本品部分经肾排出，需减量应用。
PediatricDrugs	String	儿童用药 示例值：18岁以下的患者禁用。
Interactions	String	药物相互作用 示例值：1．尿碱化剂可减低本品在尿中的溶解度，导致结晶尿和肾毒性。2．本品与茶碱类合用时茶 除半衰期(t1/2β)延长，血药浓度升高，出现茶碱中毒症状，如恶心、呕吐、震颤、不安、激动、抽 前者的血药浓度升高，必须监测环孢素血药浓度，并调整剂量。4．本品与抗凝药华法林同用时可增 小管分泌约50%，合用时可因本品血浓度增高而产生毒性。6．本品与呋喃妥因具拮抗作用，不推荐 品的吸收，建议避免合用，不能避免时在本品服药前2小时，或服药后6小时服用。8．去羟肌苷(di 合，故不宜合用。9．本品干扰咖啡因的代谢，从而导致咖啡因清除减少，血消除半衰期(t1/2β)延
ClinicalResearch	String	临床研究
PharmacologyToxicology	String	药理毒理
Pharmacokinetics	String	药代动力学 示例值：健康成人空腹一次口服本品0.4克，1～2小时血药峰浓度(Cmax)为1.32mg/ml，8小时后 相半衰期(t1/2β)为2.54小时。12小时内尿中平均排出量占给药量的27.88%，其中22.56%在6小 等的药物浓度均较高，上颚扁桃体及兔的皮肤也有分布。
Warning	String	警告
ExpireDate	String	有效期 示例值：24 月
Storage	String	贮藏 示例值：遮光，密封保存。
Pack	String	包装 示例值：铝塑泡罩包装，12片/板×2板/盒。

名称	类型	描述
Manufacturer	String	生产企业 示例值：石药集团欧意药业有限公司
ManufacturerAddress	String	生产企业地址 示例值：河北省石家庄市经济技术开发区扬子路6号
ManufacturerPhone	String	生产企业电话 示例值：0311-85900000
ManufacturerEmail	String	生产企业邮箱 示例值：info@cspc.com
ManufacturerWebsite	String	生产企业网站 示例值：http://www.cspc.com
DocRevisionTime	String	说明书制定和修订时间
References	String	参考文献
DrugDosageForm	String	剂型
DrugRoute	String	给药途径
DrugBasicCode	String	药品本位码
OctTag	String	OCT标签

SmartPredictReqData

智能预测接口请求对象

被如下接口引用：SmartPredict。

名称	类型	必选	描述
RequestCase	RequestCase	是	病历和处方信息
RequestType	Integer	是	0--默认值，同时返回**疾病预测**和**用药审查**结果 1--仅返回**疾病预测**结果 2--仅返回**用药审查**结果 已同时激活两个模块时，可按需使用 示例值：0

SmartPredictRespData

智能问诊响应数据

被如下接口引用：SmartPredict。

名称	类型	必选	描述
DiagnosisInfo	DiagnosisInfo	否	诊断辅助内容
RationalDrugInfo	RationalDrugInfo	否	用药风险信息

SuspectedDiagnosis

疑似诊断

被如下接口引用：SmartPredict。

名称	类型	必选	描述
DiseaseName	String	是	疾病名称 示例值：高血压

名称	类型	必选	描述
IcdCode	String	是	ICD代码 示例值：xxx
Symptom	String	是	症状 示例值：头晕，头痛，颈项板紧，疲劳，心悸，视物模糊，胸闷，心绞痛，气短，多汗，鼻出血
Physi	String	是	体征 示例值：心界扩大，第二心音亢进，血管杂音
Inspection	String	是	检查 示例值：心电图，肝功能，肾功能，电解质，尿常规，血脂，血糖，心脏彩超，肾脏彩超，颈动脉彩超，胸部X线，眼底动脉 影
DiseaseGuideInfo	String	是	疾病指南信息 示例值：http://127.0.0.1:8080/toolbox/AssistantDetail.html? detailid=1497901&type=disease&token=tai.MV8wXzgwMDAwMDUyOA%3D%3D.NQ%3D%3D.bb6f0c7 ff5a-11ed-8d21-2fc84f9d28c2
Probability	Float	是	置信度 示例值：0.25

SyncDepartmentData

同步科室信息

被如下接口引用：SyncDepartment。

名称	类型	必选	描述
Cmd	Integer	否	操作类型 1:获取科室列表 2:同步科室信息（增、改） 3:删除科室 示例值：1
List	Array of Department	否	科室列表

SyncDepartmentRespData

同步科室信息返回

被如下接口引用：SyncDepartment。

名称	类型	必选	描述
List	Array of Department	否	科室列表

SyncDictData

同步字典数据

被如下接口引用：SyncStandardDict。

名称	类型	必选	描述
HospitalId	String	否	医院ID 示例值：001
DictType	Integer	否	字典类型 1-给药频次 2-给药途径 3-科室 4-诊断 示例值：1
Dicts	Array of Dict	否	字典数据

TreatmentGuide

治疗方案

被如下接口引用：SmartPredict。

名称	类型	必选	描述
DoctorDiagnosis	String	是	医生诊断 示例值：急性上呼吸道感染
DiseaseName	String	是	疾病名称 示例值：急性上呼吸道感染
TreatDetailUrl	String	是	治疗详情链接 示例值：http://127.0.0.1:8080/toolbox/AssistantDetail.html?detailid=1497818&type=disease&token=tai.MV8wXzgwMDAwMDUyOA%3D%3D.NQ%3D%3D.bb6f0c7(ff5a-11ed-8d21-2fc84f9d28c2&anchor=治疗方案
TreatPlan	String	是	治疗方案 示例值：1、对症治疗 对有急性咳嗽、鼻后滴漏和咽干的病人可予以伪麻黄碱治疗以减轻鼻部充血，也可局部滴鼻应用，必要时加用解热镇痛药物，有哮喘病史忌用阿司匹林。 2、抗生素治疗 普通感冒不需要使用抗生素，有白细胞升高、咽部脓苔、咳黄痰和流鼻涕等细菌感染证据，可根据当地流行病学和经验选用口服青霉素类、第一代头孢菌素、大环内酯类药物或喹诺酮，极少需要根据病原学选用敏感抗生素。 3、抗病毒治疗 对无发热、免疫功能正常、发病不超过2天的病人一般无需应用抗病毒药物，对免疫缺陷病人，可早期常规使用。奥司他韦、利巴韦林有较广的抗病毒谱。 4、中药治疗 辩证给与清热解毒或辛温解表和有抗病毒作用的中药，有助于改善症状，缩短病程。
TreatPrinciple	String	是	治疗原则 示例值：目前无特效抗病毒药物,以对症治疗为主,同时戒烟、注意休息、多饮水、保持室内空气流通和防治继发性细菌感染。

UploadDrugData

上传药品数据

被如下接口引用：UploadDrugs。

名称	类型	必选	描述
Drugs	Array of Drug	否	药品列表

VitalSignsInfo

生命体征风险

被如下接口引用：SmartPredict。

名称	类型	必选	描述
Hit	Boolean	是	是否包含风险 示例值：true
Tips	String	是	提示

错误码

最近更新时间：2024-12-24 09:13:40

功能说明

如果返回结果中存在 Error 字段，则表示调用 API 接口失败。例如：

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

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

错误码列表

公共错误码

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

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

业务错误码

错误码	说明
AuthFailure	CAM签名/鉴权错误。