图像识别

API 文档

腾讯云

文档目录

API 文档
更新历史

最近更新时间：2025-06-19 16:42:55

第 34 次发布

发布时间：2025-06-19 16:42:45

本次发布包含了以下内容：

改善已有的文档。

删除接口：

- DetectLabelBeta

第 33 次发布

发布时间：2024-08-19 02:22:39

本次发布包含了以下内容：

改善已有的文档。

删除接口：

- DetectProductBeta

删除数据结构：

- LemmaInfo
- Location
- ProductInfo
- RegionDetected

第 32 次发布

发布时间：2024-07-05 01:19:00

本次发布包含了以下内容：

改善已有的文档。

预下线接口：

- DetectProductBeta

第 31 次发布

发布时间：2023-01-13 01:55:53

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- UpdateImage

第 30 次发布

发布时间：2022-11-22 07:01:36

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DetectChefDress
- DetectSecurity

新增数据结构：

- AttributesForBody
- BodyAttributes

第 29 次发布

发布时间：2022-11-09 06:57:46

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DetectPet

新增数据结构：

- Pet

第 28 次发布

发布时间：2022-08-31 06:53:49

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- CarPlateContent
  - 新增成员：PlateStatus, PlateStatusConfidence, PlateAngle
- CarTagItem
  - 新增成员：Orientation, OrientationConfidence

第 27 次发布

发布时间：2022-08-16 06:45:30

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- Box
  - 新增成员：CategoryId

第 26 次发布

发布时间：2022-07-29 06:16:27

本次发布包含了以下内容：

改善已有的文档。

删除接口：

- DetectCelebrity

删除数据结构：

- Face
- Labels
- Threshold

第 25 次发布

发布时间：2022-07-21 06:19:01

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- CarTagItem
  - 新增成员：PlateConfidence, TypeConfidence, ColorConfidence

第 24 次发布

发布时间：2022-07-07 06:17:32

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- ObjectInfo
  - 新增成员：AllBox

第 23 次发布

发布时间：2022-06-15 06:15:49

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DetectEnvelope

新增数据结构：

- ImageTag

第 22 次发布

发布时间：2022-06-09 06:18:37

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DetectLabelPro

第 21 次发布

发布时间：2022-06-03 06:14:26

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- CreateImage
  - 新增入参：EnableDetect, CategoryId, ImageRect
  - 新增出参：Object
- SearchImage
  - 新增入参：EnableDetect, CategoryId
  - 新增出参：Object

新增数据结构：

- Attribute
- Box
- ColorInfo
- ObjectInfo
- Rect

第 20 次发布

发布时间：2022-04-14 06:18:37

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- RecognizeCarPro

第 19 次发布

发布时间：2021-12-16 08:15:16

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- SearchImage
  - 新增入参：ImageRect

新增数据结构：

- ImageRect

第 18 次发布

发布时间：2021-11-10 08:11:42

本次发布包含了以下内容：

改善已有的文档。

新增数据结构：

- LemmaInfo

修改数据结构：

- ProductInfo
  - 新增成员：LemmaInfoList

第 17 次发布

发布时间：2021-11-09 08:16:08

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- DetectProductBeta
  - 新增入参：NeedLemma

第 16 次发布

发布时间：2021-10-29 08:10:24

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- CreateGroup
- CreateImage
- DeleteImages
- DescribeGroups
- DescribeImages
- SearchImage

新增数据结构：

- GroupInfo
- ImageInfo

第 15 次发布

发布时间：2021-09-24 10:12:47

本次发布包含了以下内容：

改善已有的文档。

新增数据结构：

- CarPlateContent

修改数据结构：

- CarTagItem

- 新增成员：PlateContent

第 14 次发布

发布时间：2021-07-01 08:08:11

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DetectLabelBeta

第 13 次发布

发布时间：2021-06-22 08:09:18

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- DetectProductBeta
  - 新增出参：ProductInfoList

第 12 次发布

发布时间：2020-06-01 08:17:59

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- DetectLabel
  - 新增出参：NewsLabels

第 11 次发布

发布时间：2020-04-23 08:16:03

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- DetectCelebrity
  - 新增出参：Threshold

新增数据结构：

- Threshold

第 10 次发布

发布时间：2020-04-10 08:15:09

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DetectProductBeta

新增数据结构：

- Location
- ProductInfo
- RegionDetected

第 9 次发布

发布时间：2020-03-31 08:13:35

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- Face
  - 新增成员：ID

第 8 次发布

发布时间：2020-02-18 14:48:02

本次发布包含了以下内容：

改善已有的文档。

删除接口：

- ImageModeration

删除数据结构：

- Candidate
- DisgustResult
- FaceRect
- FaceResult
- PoliticsResult
- PornResult
- TerrorismResult
- TextResult

第 7 次发布

发布时间：2020-01-10 10:15:14

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- CarTagItem
  - 新增成员：CarLocation

第 6 次发布

发布时间：2019-11-08 10:05:10

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- DetectLabel
  - 新增入参：Scenes
  - 新增出参：CameraLabels, AlbumLabels

修改数据结构：

- DetectLabelItem
  - 新增成员：FirstCategory, SecondCategory

第 5 次发布

发布时间：2019-11-01 11:34:51

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- CropImage
- DetectDisgust
- DetectMisbehavior

第 4 次发布

发布时间：2019-09-19 20:10:56

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- ImageModeration
  - 新增出参：TextResult

新增数据结构：

- TextResult

第 3 次发布

发布时间：2019-08-08 22:35:55

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- AssessQuality
- DetectCelebrity
- EnhanceImage

新增数据结构：

- Face
- Labels

第 2 次发布

发布时间：2019-07-25 20:32:11

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DetectProduct
- RecognizeCar

新增数据结构：

- CarTagItem
- Coord
- Product

第 1 次发布

发布时间：2019-06-13 15:41:10

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DetectLabel
- ImageModeration

新增数据结构：

- Candidate
- DetectLabelItem
- DisgustResult
- FaceRect
- FaceResult
- PoliticsResult
- PornResult
- TerrorismResult

简介

最近更新时间：2025-06-19 16:42:54

腾讯云图像分析基于深度学习等人工智能技术，提供综合性的图像智能服务，包含图像理解（解析图像中的场景、物品、动物等）、图像处理（对图像进行裁剪、美化）、图像质量评估（分析图像视觉质量）等。图像分析所使用的算法，广泛应用于腾讯内部各个产品，应用场景包含相册、信息流、社交、广告等，可以大幅提升各类产品的体验、效率。

产品功能

图像理解

- 图像标签：图像标签可以识别图片中的场景、物品等信息，包含八个大类、六十多个二级分类、数千个标签。
- 商品识别：涵盖25个大类、数百个细分类别，包括电商、广告场景下的常见商品，并输出商品所在坐标。
- 车辆识别：识别图片中车辆的车型，输出车辆品牌、车系、类型、颜色、年份等，增强版还可识别车牌信息，支持千种车型，基本覆盖市面可见的乘用车。

图像处理

- 智能裁剪：输出图片和指定的裁剪比例，可以智能识别图片的主体（包含人和物品）位置，输出推荐的裁剪坐标。可以将不同尺寸、比例的图片，快速适配不同平台、展示位的要求。
- 图像清晰度增强：通过多种智能算法，解决图片因为拍摄、滤镜、压缩导致的噪点和模糊等问题，可以用于网络图片优化、相册旧照片美化等。

图像质量评估

- 图像质量评估：评估输入图片在视觉上的质量，从多个方面评估，并同时给出综合的、客观的清晰度评分，和主观的美观度评分，广泛应用于文章封面、视频封面、相册图片去重、低质量图片过滤等。

API 概览

最近更新时间：2025-06-19 16:42:55

图像标签相关接口

接口名称	接口功能	频率限制（次/秒）
DetectLabelPro	通用图像标签	20
DetectLabel	图像标签	20

图像搜索相关接口

接口名称	接口功能	频率限制（次/秒）
CreateGroup	创建图片库	1
CreateImage	创建图片	10
DeleteImages	删除图片	10
DescribeGroups	查询图片库	20
DescribeImages	查询图片信息	20
SearchImage	检索图片	10
UpdateImage	更新图片	20

图像识别相关接口

接口名称	接口功能	频率限制（次/秒）
DetectProduct	商品识别	20
RecognizeCarPro	车辆识别（增强版）	20
RecognizeCar	车辆识别	20
DetectEnvelope	文件封识别	20
DetectChefDress	厨师穿戴识别接口	2
DetectPet	宠物识别	20
DetectSecurity	安全属性识别	2

图像处理相关接口

接口名称	接口功能	频率限制（次/秒）
AssessQuality	图像质量评估	20
CropImage	图片智能裁剪	20
EnhanceImage	图像清晰度增强	20

图像审核相关接口

接口名称	接口功能	频率限制（次/秒）
DetectDisgust	恶心检测	20
DetectMisbehavior	不良行为识别	20

注意：

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

调用方式
请求结构

最近更新时间：2025-06-19 16:42:54

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

注意：由于金融区和非金融区是隔离不互通的，因此当访问金融区服务时（公共参数 Region 为金融区地域），需要同时指定带金融区地域的域名，最好和 Region 的地域保持一致。

金融区接入地域	金融区域名
华东地区（上海金融）	tiia.ap-shanghai-fsi.tencentcloudapi.com

金融区接入地域	金融区域名
华南地区（深圳金融）	tiia.ap-shenzhen-fsi.tencentcloudapi.com

2. 通信协议

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

3. 请求方法

支持的 HTTP 请求方法：

- POST（推荐）
- GET

POST 请求支持的 Content-Type 类型：

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

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

4. 字符编码

均使用 UTF-8 编码。

公共参数

最近更新时间：2024-12-23 11:03:54

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 tiia；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, Sig
```

```
nedHeaders=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, Sig
nedHeaders=content-type;host, Signature=582c400e06b5924a6f2b5d7d672d79c15b13162d9279b0855cfba6789a8edb4c
Content-Type: application/json
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1527672334
X-TC-Region: ap-guangzhou

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

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

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

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

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

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

10
--58731222010402--
```

使用签名方法 v1 的公共参数

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

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

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

HTTP GET 请求结构示例：

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

Host: cvm.tencentcloudapi.com
```

HTTP POST 请求结构示例：

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

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

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

地域列表

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

地域	取值
华北地区（北京）	ap-beijing
华南地区（广州）	ap-guangzhou
华东地区（上海）	ap-shanghai

签名方法 v3

最近更新时间：2024-12-25 02:04:07

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=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b4609
3f6ab6c4f" \
-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。

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

根据以上规则，示例中得到的规范请求串如下：

```
POST
/

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

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

2. 拼接待签名字符串

按如下格式拼接待签名字符串：

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

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

字段名称	解释
	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 = "Gu5t9xGARNpq86cd98joQYCN3Cozk1qA"
SecretDate = HMAC_SHA256("TC3" + SecretKey, Date)
SecretService = HMAC_SHA256(SecretDate, Service)
SecretSigning = HMAC_SHA256(SecretService, "tc3_request")
```

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

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

字段名称	解释
SecretKey	原始的 SecretKey，即 Gu5t9xGARNpq86cd98joQYCN3Cozk1qA。
Date	即 Credential 中的 Date 字段信息。此示例取值为 2019-02-25。
Service	即 Credential 中的 Service 字段信息。此示例取值为 cvm。

2）计算签名，伪代码如下：

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

此示例计算结果是 10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f 。

4. 拼接 Authorization

按如下格式拼接 Authorization：

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

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

根据以上规则，示例中得到的值为：

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

最终完整的调用信息如下：

```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2019-02-25/cvm/tc3_request, Sig
nedHeaders=content-type;host;x-tc-action, Signature=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6
ab6c4f
Content-Type: application/json; charset=utf-8
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1551113065
X-TC-Region: ap-guangzhou

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

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

签名演示

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

- Python

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

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

- Signature Demo

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

Java

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

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

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

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

public static void main(String[] args) throws Exception {
String service = "cvm";
String host = "cvm.tencentcloudapi.com";
```

```
String region = "ap-guangzhou";
String action = "DescribeInstances";
String version = "2017-03-12";
String algorithm = "TC3-HMAC-SHA256";
String timestamp = "1551113065";
//String timestamp = String.valueOf(System.currentTimeMillis() / 1000);
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
// 注意时区，否则容易出错
sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
String date = sdf.format(new Date(Long.valueOf(timestamp + "000")));

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

String payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"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";
```

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

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

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

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

$curl = "curl -X POST https://".$host
.' -H "Authorization: '.$authorization.'"'
```

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

Ruby

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

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

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

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

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

DotNet

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

public class Application
{
public static string SHA256Hex(string s)
```

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

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

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

// ************* 步骤 2：拼接待签名字符串 *************
string credentialScope = datestr + "/" + service + "/" + "tc3_request";
string hashedCanonicalRequest = SHA256Hex(canonicalRequest);
```

```
string stringToSign = algorithm + "\n"
+ requestTimestamp.ToString() + "\n"
+ credentialScope + "\n"
+ hashedCanonicalRequest;
Console.WriteLine(stringToSign);

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

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

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

```
stance-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" + hashedCanonicalReq
uest;
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\": \"inst
ance-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）。

返回结果

最近更新时间：2024-03-12 01:55:37

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

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

正确返回结果

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

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

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

错误返回结果

若调用失败，其返回值示例如下为：

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

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

公共错误码

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

参数类型

最近更新时间：2022-08-10 06:53:43

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。

图像标签相关接口
通用图像标签

最近更新时间：2025-06-19 16:42:51

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

通用图像标签可识别数千种常见物体或场景，覆盖日常物品、场景、动物、植物、食物、饮品、交通工具等多个大类，返回主体的标签名称和所属细分类目等。

- 可前往 图像标签 产品文档中查看更多产品信息。
- 公共参数中的签名方式必须指定为V3版本，即配置SignatureMethod参数为TC3-HMAC-SHA256。

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：DetectLabelPro。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
ImageUrl	否	String	图片 URL 地址。 图片限制： • 图片格式：PNG、JPG、JPEG、BMP。 • 图片大小：所下载图片经Base64编码后不超过4M。图片下载时间不超过3秒。 建议： • 图片像素：大于50*50像素，否则影响识别效果； • 长宽比：长边:短边<5； • 接口响应时间会受到图片下载时间的影响，建议使用更可靠的存储服务，推荐将图片存储在腾讯云COS。 示例值：https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg
ImageBase64	否	String	图片 Base64 编码数据。 与ImageUrl同时存在时优先使用ImageUrl字段。 图片限制： • 图片格式：PNG、JPG、JPEG、BMP。 • 图片大小：经Base64编码后不超过4M。 **注意：图片需要Base64编码，并且要去掉编码头部。** 示例值：/9j/4AAQSkZJRgABAQAAAQABAAD/4gIo...IftXF/DjFZNXoSP5V2U0HMt/1FQf/Z

3. 输出参数

参数名称	类型	描述
Labels	Array of DetectLabelItem	返回标签数组。 示例值：[{"Name": "塔楼","FirstCategory": "场景","SecondCategory": "建筑","Confidence": 81},{"Name": "夜晚","FirstCategory": "场景","SecondCategory": "自然风光","Confidence": 79}]
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 图像标签检测请求失败

输入示例

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

{
"ImageUrl": "https://cos.ap-singapore.myqcloud.com/input.png"
}
```

输出示例

```
{
"Response": {
"Error": {
"Code": "FailedOperation.DownLoadError",
"Message": "下载失败"
},
"RequestId": "a169390a-6ff3-4c42-ad25-a7858c35e576"
}
}
```

示例2 图像标签检测请求成功

输入示例

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

{
"ImageUrl": "https://cos.ap-singapore.myqcloud.com/input.png"
}
```

输出示例


```
{
"Response": {
"Labels": [
{
"Name": "塔楼",
"FirstCategory": "场景",
"SecondCategory": "建筑",
"Confidence": 81
},
{
"Name": "夜晚",
"FirstCategory": "场景",
"SecondCategory": "自然风光",
"Confidence": 79
},
{
"Name": "天际线",
"FirstCategory": "场景",
"SecondCategory": "自然风光",
"Confidence": 77
},
{
"Name": "城市景观",
"FirstCategory": "场景",
"SecondCategory": "其他",
"Confidence": 77
},
{
"Name": "城市",
"FirstCategory": "场景",
"SecondCategory": "生活/娱乐场所",
"Confidence": 72
},
{
"Name": "都市",
"FirstCategory": "场景",
"SecondCategory": "其他",
"Confidence": 69
}
],
"RequestId": "e6d685b1-d898-4dc9-a545-cfeb3a988da8"
}
}
```


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.InvalidAuthorization	认证失败。
FailedOperation.DownloadError	文件下载错误。
FailedOperation.ImageDecodeFailed	图片解码失败。
FailedOperation.ImageDownloadError	图片下载错误。
FailedOperation.ImageNotSupported	不支持的图片文件。
FailedOperation.ImageUnQualified	图片不满足检测要求。
FailedOperation.InvokeChargeError	调用计费返回失败。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
FailedOperation.TooLargeFileError	文件太大。
FailedOperation.UnKnowError	内部错误。
FailedOperation.Unknown	未知错误。
InvalidParameterValue.InvalidParameterValueLimit	参数值错误。
LimitExceeded.TooLargeFileError	文件太大。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。
ResourcesSoldOut.ChargeStatusException	计费状态异常。

图像标签

最近更新时间：2025-06-19 16:42:51

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

图像标签利用深度学习技术，可以对图片进行智能分类、物体识别等。

目前支持八个大类、六十多个子类、数千个标签，涵盖各种日常场景、动植物、物品、美食等。

图像标签提供四个版本供选择：

- 摄像头版：针对搜索、手机摄像头照片进行优化，涵盖大量卡证、日常物品、二维码条形码。
- 相册版：针对手机相册、网盘进行优化，去除相册和网盘中不常见的标签，针对相册常见图片类型（人像、日常活动、日常物品等）识别效果更好。
- 网络版：针对网络图片进行优化，涵盖标签更多，满足长尾识别需求。
- 新闻版：针对新闻、资讯、广电等行业进行优化，增加定制识别，支持万级图像标签。

为了方便使用、减少图片传输次数，图像标签将不同版本包装成多合一接口，实际上是多个服务，分别计费。建议在接入初期，对四个版本进行对比评估后选择合适的版本使用。

> 说明：
>
> - 图像标签已升级服务，建议使用新版接口通用图像标签。
> - 图像标签摄像头版、相册版、网络版、新闻版分别按照各自的实际使用次数进行收费，例如一张图片同时使用相册版、摄像头版，则按照两次调用计费。建议测试对比后从中选择一个最合适的版本使用即可。

- 可前往 图像标签 产品文档中查看更多产品信息。
- 公共参数中的签名方式必须指定为V3版本，即配置SignatureMethod参数为TC3-HMAC-SHA256。

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：DetectLabel。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
ImageBase64	否	String	图片 Base64 编码数据。 与ImageUrl同时存在时优先使用ImageUrl字段。 图片限制： • 图片格式：PNG、JPG、JPEG、BMP。 • 图片大小：经Base64编码后不超过4M。

参数名称	必选	类型	描述
			注意：图片需要Base64编码，并且要去掉编码头部。 示例值：/9j/4AAQSkZJRgABAQAAAQABAAD/4glo...IftXF/DjFZNXoSP5V2U0HMt/1FQf/Z
ImageUrl	否	String	图片 URL 地址。 图片限制： • 图片格式：PNG、JPG、JPEG、BMP。 • 图片大小：所下载图片经Base64编码后不超过4M。图片下载时间不超过3秒。 建议： • 图片像素：大于50*50像素，否则影响识别效果； • 长宽比：长边:短边<5； • 接口响应时间会受到图片下载时间的影响，建议使用更可靠的存储服务，推荐将图片存储在腾讯云COS。 示例值：https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg
Scenes.N	否	Array of String	本次调用支持的识别场景，可选值如下： • WEB，针对网络图片优化； • CAMERA，针对手机摄像头拍摄图片优化； • ALBUM，针对手机相册、网盘产品优化； • NEWS，针对新闻、资讯、广电等行业优化； 如果不传此参数，则默认为WEB。 支持多场景（Scenes）一起检测。例如，使用 Scenes=["WEB", "CAMERA"]，即对一张图片使用两个模型同时检测，输出两套识别结果。 示例值：["WEB","CAMERA","ALBUM"]

3. 输出参数

参数名称	类型	描述
Labels	Array of DetectLabelItem	Web网络版标签结果数组。如未选择WEB场景，则为空。 注意：此字段可能返回 null，表示取不到有效值。 示例值：[{"Name": "塔楼","FirstCategory": "场景","SecondCategory": "建筑","Confidence": 81}]
CameraLabels	Array of DetectLabelItem	Camera摄像头版标签结果数组。如未选择CAMERA场景，则为空。 注意：此字段可能返回 null，表示取不到有效值。 示例值：[{"Name": "夜景","FirstCategory": "场景","SecondCategory": "自然风光","Confidence": 92}]
AlbumLabels	Array of DetectLabelItem	Album相册版标签结果数组。如未选择ALBUM场景，则为空。 注意：此字段可能返回 null，表示取不到有效值。 示例值：[{"Name": "夜景","FirstCategory": "场景","SecondCategory": "自然风光","Confidence": 93},{"Name": "塔","FirstCategory": "场景","SecondCategory": "建筑","Confidence": 82}]
NewsLabels	Array of DetectLabelItem	News新闻版标签结果数组。如未选择NEWS场景，则为空。 新闻版目前为测试阶段，暂不提供每个标签的一级、二级分类信息的输出。 注意：此字段可能返回 null，表示取不到有效值。 示例值：[{"Confidence": 80, "Name": "塔楼", "SecondCategory": "建筑", "FirstCategory": "场景"}]
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 图像标签检测请求失败

输入示例

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

{
"ImageUrl": "https://cos.ap-singapore.myqcloud.com/input.png"
}
```

输出示例

```
{
"Response": {
"Error": {
"Code": "FailedOperation.DownLoadError",
"Message": "下载失败"
},
"RequestId": "a169390a-6ff3-4c42-ad25-a7858c35e576"
}
}
```

示例2 图像标签检测请求成功

输入示例

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

{
"ImageUrl": "https://cos.ap-singapore.myqcloud.com/input.png",
"Scenes": [
"WEB",
"CAMERA",
"ALBUM"
]
}
```

输出示例

```
{
"Response": {
"NewsLabels": [
{
"Confidence": 80,
"Name": "塔楼",
```

```
"SecondCategory": "建筑",
"FirstCategory": "场景"
}
],
"Labels": [
{
"Name": "塔楼",
"FirstCategory": "场景",
"SecondCategory": "建筑",
"Confidence": 81
},
{
"Name": "夜晚",
"FirstCategory": "场景",
"SecondCategory": "自然风光",
"Confidence": 79
},
{
"Name": "天际线",
"FirstCategory": "场景",
"SecondCategory": "自然风光",
"Confidence": 77
},
{
"Name": "城市景观",
"FirstCategory": "场景",
"SecondCategory": "其他",
"Confidence": 77
},
{
"Name": "城市",
"FirstCategory": "场景",
"SecondCategory": "生活/娱乐场所",
"Confidence": 72
},
{
"Name": "都市",
"FirstCategory": "场景",
"SecondCategory": "其他",
"Confidence": 69
}
],
"CameraLabels": [
{
"Name": "夜景",
"FirstCategory": "场景",
"SecondCategory": "自然风光",
"Confidence": 92
},
{
"Name": "城市",
```

```
"FirstCategory": "场景",
"SecondCategory": "建筑",
"Confidence": 3
},
{
"Name": "游乐园",
"FirstCategory": "场景",
"SecondCategory": "生活/娱乐场所",
"Confidence": 2
},
{
"Name": "大厦",
"FirstCategory": "场景",
"SecondCategory": "建筑",
"Confidence": 1
},
{
"Name": "桥",
"FirstCategory": "场景",
"SecondCategory": "建筑",
"Confidence": 0
}
],
"AlbumLabels": [
{
"Name": "夜景",
"FirstCategory": "场景",
"SecondCategory": "自然风光",
"Confidence": 93
},
{
"Name": "塔",
"FirstCategory": "场景",
"SecondCategory": "建筑",
"Confidence": 82
},
{
"Name": "城市",
"FirstCategory": "场景",
"SecondCategory": "建筑",
"Confidence": 5
}
],
"RequestId": "e6d685b1-d898-4dc9-a545-cfeb3a988da8"
}
}
```


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.InvalidAuthorization	认证失败。
FailedOperation.BalanceInsufficient	余额不足，开通失败，请充值后再开通。
FailedOperation.DownloadError	文件下载错误。
FailedOperation.ImageDecodeFailed	图片解码失败。
FailedOperation.ImageDownloadError	图片下载错误。
FailedOperation.ImageNotSupported	不支持的图片文件。
FailedOperation.ImageUnQualified	图片不满足检测要求。
FailedOperation.InvokeChargeError	调用计费返回失败。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
FailedOperation.UnKnowError	内部错误。
FailedOperation.Unknown	未知错误。
InvalidParameterValue.InvalidParameterValueLimit	参数值错误。
LimitExceeded.TooLargeFileError	文件太大。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。
ResourcesSoldOut.ChargeStatusException	计费状态异常。

图像搜索相关接口
创建图片库

最近更新时间：2025-06-19 16:42:52

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

本接口用于创建一个空的图片库，图片库主要用于存储在创建图片时提取的图片特征数据，如果图片库已存在则返回错误。不同的图片库类型对应不同的图像搜索服务类型，根据输入参数GroupType区分。

服务类型	GroupType	功能描述
通用图像搜索	4	通用图像搜索1.0版。 在自建图片库中搜索相同原图或相似图片集，并给出相似度打分，可支持裁剪、翻转、调色、加水印等二次编辑后的图片搜索。适用于图片版权保护、原图查询等场景。
商品图像搜索	8	商品图像搜索3.0升级版（推荐）。 在自建图库中搜索同款或相似商品，并给出相似度打分。对于服饰类商品可支持识别服饰类别、属性等信息。适用于商品分类、检索、推荐等电商场景。
	7	商品图像搜索2.0版。 功能和3.0升级版类似。
	5	商品图像搜索1.0版。 功能和3.0升级版类似。
图案花纹搜索	6	图案花纹搜索1.0版。 在自建图库中搜索相似的图案、logo、纹理等图像元素或主体，并给出相似度打分。

- 可前往 图像搜索 产品文档中查看更多产品信息。

默认接口请求频率限制：1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：CreateGroup。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
GroupId	是	String	图库ID，不可重复，仅支持字母、数字和下划线。图库数量单个用户上限为30。 示例值：test_group
GroupName	是	String	图库名称描述。 示例值：示例图库

参数名称	必选	类型	描述
MaxCapacity	是	Integer	图片库可容纳的最大图片特征条数，一张图片对应一条图片特征数据，不支持修改。 单个图片库容量最大可达亿级，达到容量限制后继续创建图片将会报错。 注意，包月计费下支持绑定的最小库容量为500万。 示例值：100
Brief	否	String	图库简介。 示例值：示例用图库
MaxQps	否	Integer	访问限制默认为10qps，如需扩容请联系在线客服申请。 示例值：10
GroupType	否	Integer	图库类型，用于决定图像搜索的服务类型和算法版本，默认为4。 GroupType不支持修改，若不确定适用的服务类型，建议先对不同类型分别小规模测试后再开始正式使用。 参数取值： 4：通用图像搜索1.0版。 8：商品图像搜索3.0升级版。 7：商品图像搜索2.0版。 5：商品图像搜索1.0版。 6：图案花纹搜索1.0版。 1 – 3：通用图像搜索旧版，不推荐使用。 示例值：1

3. 输出参数

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

4. 示例

示例1 调用成功示例

调用成功示例

输入示例

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

{
"Brief": "brief_test",
"GroupName": "name_test",
"MaxQps": "10",
"MaxCapacity": "100",
"GroupId": "test_group"
}
```

输出示例

```
{
"Response": {
"RequestId": "7ba53193-90af-4440-b706-b3d3a0657982"
}
}
```

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.InvalidAuthorization	认证失败。
FailedOperation.GroupCountExceeded	图库数量超出限制。
FailedOperation.InnerError	内部错误。
FailedOperation.ParameterEmpty	参数为空。
FailedOperation.RequestError	后端服务请求失败。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
FailedOperation.UnKnowError	内部错误。
FailedOperation.Unknown	未知错误。
InvalidParameter.InvalidParameter	参数取值错误。
InvalidParameterValue.BriefTooLong	图库简介过长。

错误码	描述
InvalidParameterValue.ImageGroupIdAlreadyExist	图库ID已存在。
InvalidParameterValue.ImageGroupIdIllegal	图库ID不合法。
InvalidParameterValue.ImageGroupIdNotExist	图库ID不存在。
InvalidParameterValue.ImageGroupIdTooLong	图库ID超出长度限制。
InvalidParameterValue.ImageGroupNameEmpty	图库名称为空。
InvalidParameterValue.ImageGroupNameTooLong	图库名称超出长度限制。
InvalidParameterValue.InvalidParameterValueLimit	参数值错误。
InvalidParameterValue.LimitExceed	返回数量不在合法范围内。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。

创建图片

最近更新时间：2025-06-19 16:42:52

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

创建图片，并添加对应图片的自定义信息。模型将在创建图片时自动提取图像特征并存储到指定的图片库中，每创建一张图片会对应提取和存储一条图片特征数据。

- 可前往 图像搜索 产品文档中查看更多产品信息。

默认接口请求频率限制：10次/秒。

推荐使用 API Explorer　点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：CreateImage。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
GroupId	是	String	图库ID。 示例值：test_group
EntityId	是	String	物品ID，最多支持64个字符。 一个物品ID可以包含多张图片，若EntityId已存在，则对其追加图片。同一个EntityId，最大支持10张图。 示例值：test_entity
PicName	是	String	图片名称，最多支持64个字符， PicName唯一确定一张图片，具有唯一性。 示例值：new_pic
ImageUrl	否	String	图片的 Url 。对应图片 base64 编码后大小不可超过5M。 ImageUrl和ImageBase64必须提供一个，如果都提供，只使用ImageUrl。 图片限制： • 图片格式：支持PNG、JPG、JPEG、BMP，不支持 GIF 图片。 • 图片大小：对应图片 base64 编码后大小不可超过5M。图片分辨率不超过4096*4096。 • 如果在商品图像搜索中开启主体识别，分辨率不超过2000*2000，图片长宽比小于10。 建议： • 图片存储于腾讯云的Url可保障更高下载速度和稳定性，建议图片存储于腾讯云。非腾讯云存储的Url速度和稳定性可能受一定影响。 示例值：https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg
CustomContent	否	String	图片自定义备注内容，最多支持4096个字符，查询时原样带回。 示例值：备注

参数名称	必选	类型	描述
ImageBase64	否	String	图片 base64 数据，base64 编码后大小不可超过5M。 图片限制： • 图片格式：支持PNG、JPG、JPEG、BMP，不支持 GIF 图片。 • 图片大小：base64 编码后大小不可超过5M。图片分辨率不超过4096*4096。 • 如果在商品图像搜索中开启主体识别，分辨率不超过2000*2000，图片长宽比小于10。 示例 值：/9j/4AAQSkZJRgABAQAAAQABAAD/4glo...lftXF/DjFZNXoSP5V2U0HMt/1FQf/Z
Tags	否	String	图片自定义标签，最多不超过10个，格式为JSON。 示例值：{"test":"1"}
EnableDetect	否	Boolean	是否需要启用主体识别，默认为TRUE。 • 为TRUE时，启用主体识别，返回主体信息。若没有指定ImageRect，自动提取最大面积主体创建图片并进行主体识别。主体识别结果可在Response中获取。 • 为FALSE时，不启用主体识别，不返回主体信息。若没有指定ImageRect，以整张图创建图片。 **注意：仅服务类型为商品图像搜索时才生效。** 示例值：true
CategoryId	否	Integer	图像类目ID。 若设置类目ID，提取以下类目的主体创建图片。 类目取值说明： 0：上衣。 1：裙装。 2：下装。 3：包。 4：鞋。 5：配饰。 **注意：仅服务类型为商品图像搜索时才生效。** 示例值：1
ImageRect	否	Rect	图像主体区域。 若设置主体区域，提取指定的区域创建图片。 示例值：{x:100,y:200,width:200,height:200}

3. 输出参数

参数名称	类型	描述
Object	ObjectInfo	输入图的主体信息。 若启用主体识别且在请求中指定了类目ID或主体区域，以指定的主体为准。若启用主体识别且没有指定，以最大面积主体为准。 **注意：仅服务类型为商品图像搜索时才生效。** 注意：此字段可能返回 null，表示取不到有效值。 示例值：{"CategoryId":0}
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 调用失败示例

调用失败示例

输入示例

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

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

{
"GroupId": "test_group",
"EntityId": "test-entity",
"PicName": "test_anme",
"ImageUrl": "https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input"
}
```

输出示例

```
{
"Response": {
"Error": {
"Code": "FailedOperation.DownloadError",
"Message": "文件下载失败。"
},
"RequestId": "258fb494-54db-4814-80e8-b4d2b75c5c83"
}
}
```

示例2 调用成功示例

调用成功示例

输入示例

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

{
"GroupId": "test_group",
"EntityId": "test-entity",
"PicName": "test_anme",
"ImageUrl": "https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg"
}
```

输出示例

```
{
"Response": {
"Object": null,
"RequestId": "49120671-d169-4714-a212-cf755e49fad8"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.DownloadError	文件下载错误。
FailedOperation.EmptyImageError	图片内容为空。
FailedOperation.ImageDecodeFailed	图片解码失败。
FailedOperation.ImageDownloadError	图片下载错误。
FailedOperation.ImageEntityCountExceed	超出Entity数量限制。
FailedOperation.ImageGroupChargeStatusClose	停止服务，控制台开关关闭。
FailedOperation.ImageNotFoundInfo	未找到图片信息。
FailedOperation.ImageNotSupported	不支持的图片文件。
FailedOperation.ImageNumExceed	超出图库限制。
FailedOperation.ImageResolutionExceed	图片分辨率过大。
FailedOperation.ImageSizeExceed	base64编码后的图片数据过大。
FailedOperation.ImageUrlInvalid	url地址不合法，无法下载。
FailedOperation.InnerError	内部错误。
FailedOperation.RequestError	后端服务请求失败。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。

错误码	描述
FailedOperation.ServerError	算法服务异常，请重试。
FailedOperation.UnKnowError	内部错误。
FailedOperation.Unknown	未知错误。
InvalidParameter.ImageFormatNotSupport	不支持的图片格式。
InvalidParameter.InvalidParameter	参数取值错误。
InvalidParameter.PictureSolidColorError	图片不可为纯色图。
InvalidParameterValue.CustomContentTooLong	自定义内容过长。
InvalidParameterValue.EntityIdEmpty	物品ID为空。
InvalidParameterValue.EntityIdTooLong	物品ID超出长度限制。
InvalidParameterValue.ImageGroupIdIllegal	图库ID不合法。
InvalidParameterValue.ImageGroupIdNotExist	图库ID不存在。
InvalidParameterValue.ImageGroupIdTooLong	图库ID超出长度限制。
InvalidParameterValue.PicNameAlreadyExist	图片名称重复。
InvalidParameterValue.PicNameEmpty	图片名称为空。
InvalidParameterValue.PicNameInvalid	PicName不合法。
InvalidParameterValue.PicNameTooLong	图片名称超出长度限制。
InvalidParameterValue.TagsKeysExceed	标签数量过多。
InvalidParameterValue.TagsValueIllegal	标签值类型不合法。
InvalidParameterValue.TagsValueSizeExceed	标签值长度过长。
MissingParameter.ErrorParameterEmpty	必选参数为空。
RequestLimitExceeded	请求的次数超过了频率限制。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.IsOpening	服务正在开通中，请稍等。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。

删除图片

最近更新时间：2025-06-19 16:42:52

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

删除图片。

- 可前往 图像搜索 产品文档中查看更多产品信息。

默认接口请求频率限制：10次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DeleteImages。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
GroupId	是	String	图库名称。 示例值：test_group
EntityId	是	String	物品ID。 示例值：4170
PicName	否	String	图片名称，如果不指定本参数，则删除EntityId下所有的图片；否则删除指定的图。 示例值：my_pic

3. 输出参数

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

4. 示例

示例1 调用失败示例

输入示例

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

{
"GroupId": "test_group",
"EntityId": "7263",
"PicName": "test_pic"
}
```

输出示例

```
{
"Response": {
"Error": {
"Code": "InvalidParameter.InvalidParameter",
"Message": "参数不合法。"
},
"RequestId": "cac925c7-c939-4eb6-8db5-d642f7385c30"
}
}
```

示例2 调用成功示例

输入示例

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

{
"GroupId": "my_group",
"EntityId": "7263",
"PicName": "test_pic"
}
```

输出示例

```
{
"Response": {
"RequestId": "cac925c7-c939-4eb6-8db5-d642f7385c30"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.ImageDeleteFailed	图片删除失败。
FailedOperation.ImageNotFoundInfo	未找到图片信息。
FailedOperation.InnerError	内部错误。
FailedOperation.RequestError	后端服务请求失败。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
FailedOperation.Unknown	未知错误。
InvalidParameterValue.EntityIdEmpty	物品ID为空。
InvalidParameterValue.EntityIdTooLong	物品ID超出长度限制。
InvalidParameterValue.ImageGroupIdIllegal	图库ID不合法。
InvalidParameterValue.ImageGroupIdNotExist	图库ID不存在。
InvalidParameterValue.ImageGroupIdTooLong	图库ID超出长度限制。
InvalidParameterValue.PicNameInvalid	PicName不合法。
InvalidParameterValue.PicNameTooLong	图片名称超出长度限制。
RequestLimitExceeded	请求的次数超过了频率限制。
ResourceUnavailable.InArrears	账号已欠费。

错误码	描述
ResourceUnavailable.IsOpening	服务正在开通中，请稍等。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。

查询图片库

最近更新时间：2025-06-19 16:42:52

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

查询所有的图库信息。

- 可前往 图像搜索 产品文档中查看更多产品信息。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeGroups。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
Offset	否	Integer	起始序号，默认值为0。 示例值：0
Limit	否	Integer	返回数量，默认值为10，最大值为100。 示例值：10
GroupId	否	String	图库ID，如果不为空，则返回指定库信息。 示例值：test_group

3. 输出参数

参数名称	类型	描述
Groups	Array of GroupInfo	图库信息
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 调用成功示例

查询当前用户的图片库信息。

输入示例

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

{
"Offset": 0,
"Limit": 100
}
```

输出示例

```
{
"Response": {
"Groups": [
{
"Brief": "简介",
"CreateTime": "2023-02-02 18:04:08",
"GroupId": "group_test_id",
"GroupName": "group_test_id_name",
"GroupType": 4,
"MaxCapacity": 10000,
"MaxQps": 10,
"PicCount": 1,
"UpdateTime": "2023-02-02 18:04:41"
}
],
"RequestId": "347a9402-b68c-42d0-8a75-d1b123fe83d5"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.RequestError	后端服务请求失败。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
InvalidParameter.InvalidParameter	参数取值错误。
InvalidParameterValue.ImageGroupIdIllegal	图库ID不合法。
InvalidParameterValue.ImageGroupIdNotExist	图库ID不存在。
InvalidParameterValue.ImageGroupIdTooLong	图库ID超出长度限制。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。

查询图片信息

最近更新时间：2025-06-19 16:42:52

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

获取指定图片库中的图片列表。

- 可前往 图像搜索 产品文档中查看更多产品信息。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeImages。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
GroupId	是	String	图库名称。 示例值：test_group
EntityId	是	String	物品ID。 示例值：test_entity
PicName	否	String	图片名称。 示例值：pic_name

3. 输出参数

参数名称	类型	描述
GroupId	String	图库名称。 示例值：work
EntityId	String	物品ID。 示例值：test3
ImageInfos	Array of ImageInfo	图片信息。
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 调用成功示例

输入示例

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

{
"EntityId": "test3",
"GroupId": "work"
}
```

输出示例

```
{
"Response": {
"EntityId": "test3",
"GroupId": "work",
"ImageInfos": [
{
"CustomContent": "custom",
"EntityId": "test3",
"PicName": "4000",
"Score": 0,
"Tags": "{\"value\": 20}"
},
{
"CustomContent": "custom",
"EntityId": "test3",
"PicName": "4001",
"Score": 0,
"Tags": "{\"value\": 20}"
},
{
"CustomContent": "custom",
"EntityId": "test3",
"PicName": "4002",
"Score": 0,
"Tags": "{\"value\": 20}"
}
],
"RequestId": "65996842-8bb5-430f-bb40-460b3c73db76"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.ImageNotFoundInfo	未找到图片信息。
FailedOperation.RequestError	后端服务请求失败。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
InvalidParameterValue.EntityIdEmpty	物品ID为空。
InvalidParameterValue.EntityIdTooLong	物品ID超出长度限制。
InvalidParameterValue.ImageGroupIdIllegal	图库ID不合法。
InvalidParameterValue.ImageGroupIdNotExist	图库ID不存在。
InvalidParameterValue.ImageGroupIdTooLong	图库ID超出长度限制。
InvalidParameterValue.PicNameEmpty	图片名称为空。
InvalidParameterValue.PicNameTooLong	图片名称超出长度限制。
RequestLimitExceeded	请求的次数超过了频率限制。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。
ResourceUnavailable.StopUsing	帐号已停服。

检索图片

最近更新时间：2025-06-19 16:42:51

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

本接口用于对一张图片，在指定图片库中检索出与之相似的图片列表。

- 可前往 图像搜索 产品文档中查看更多产品信息。

默认接口请求频率限制：10次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：SearchImage。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
GroupId	是	String	图库名称。 示例值：test_group
ImageUrl	否	String	图片的 Url 。 ImageUrl和ImageBase64必须提供一个，如果都提供，只使用ImageUrl。 图片限制： • 图片格式：支持PNG、JPG、JPEG、BMP，不支持 GIF 图片。 • 图片大小：对应图片 base64 编码后大小不可超过5M。图片分辨率不超过4096*4096。 • 如果在商品图像搜索中开启主体识别，分辨率不超过2000*2000，图片长宽比小于10。 建议： • 图片存储于腾讯云的Url可保障更高下载速度和稳定性，建议图片存储于腾讯云。非腾讯云存储的Url速度和稳定性可能受一定影响。 示例值：https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg
ImageBase64	否	String	图片 base64 数据。 ImageUrl和ImageBase64必须提供一个，如果都提供，只使用ImageUrl。 图片限制： • 图片格式：支持PNG、JPG、JPEG、BMP，不支持 GIF 图片。 • 图片大小：base64 编码后大小不可超过5M。图片分辨率不超过4096*4096。 • 如果在商品图像搜索中开启主体识别，分辨率不超过2000*2000，图片长宽比小于10。 示例 值：/9j/4AAQSkZJRgABAQAAAQABAAD/4gIo...IftXF/DjFZNXoSP5V2U0HMt/1FQf/Z
Limit	否	Integer	返回结果的数量，默认值为10，最大值为100。 按照相似度分数由高到低排序。

参数名称	必选	类型	描述
			服务类型为图案花纹搜索时Limit = 1，最多只能返回1个结果。 示例值：10
Offset	否	Integer	返回结果的起始序号，默认值为0。 示例值：0
MatchThreshold	否	Integer	匹配阈值。 只有图片相似度分数超过匹配阈值的结果才会返回。 当MatchThreshold为0（默认值）时，各服务类型将按照以下默认的匹配阈值进行结果过滤： • 通用图像搜索1.0版：50。 • 商品图像搜索2.0升级版：45。 • 商品图像搜索1.0版：28。 • 图案花纹搜索1.0版：56。 建议： 可以手动调整MatchThreshold值来控制输出结果的范围。如果发现无检索结果，可能是因为图片相似度较低导致检索结果被匹配阈值过滤，建议调整为较低的阈值后再次尝试检索。 示例值：0
Filter	否	String	标签过滤条件。 针对创建图片时提交的Tags信息进行条件过滤。支持>、>=、<、<=、=，!=，多个条件之间支持AND和OR进行连接。 最大支持64字符。 示例值：value > 1
ImageRect	否	ImageRect	图像主体区域。 若设置主体区域，提取指定的区域进行检索。 示例值：{"X":231,"Y":322,"Width":"200","Height":200}
EnableDetect	否	Boolean	是否需要启用主体识别，默认为TRUE 。 • 为TRUE时，启用主体识别，返回主体信息。若没有指定ImageRect，自动提取最大面积主体进行检索并进行主体识别。主体识别结果可在Response中获取。 • 为FALSE时，不启用主体识别，不返回主体信息。若没有指定ImageRect，以整张图检索图片。 **注意：仅服务类型为商品图像搜索时才生效。** 示例值：true
CategoryId	否	Integer	图像类目ID。 若设置类目ID，提取以下类目的主体进行检索。 类目取值说明： 0：上衣。 1：裙装。 2：下装。 3：包。 4：鞋。 5：配饰。 **注意：仅服务类型为商品图像搜索时才生效。** 示例值：1

3. 输出参数

参数名称	类型	描述
Count	Integer	返回结果数量。 示例值：1
ImageInfos	Array of ImageInfo	图片信息。 注意：此字段可能返回 null，表示取不到有效值。 示例值：[{"CustomContent":"备注","EntityId":"work-1","PicName":"work-1-1","Score":100,"Tags":"{}"}]

参数名称	类型	描述
Object	ObjectInfo	输入图的主体信息。 若启用主体识别且在请求中指定了类目ID或主体区域，以指定的主体为准。若启用主体识别且没有指定，以最大面积主体为准。 **注意：仅服务类型为商品图像搜索时才生效。** 注意：此字段可能返回 null，表示取不到有效值。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 调用成功示例

输入示例

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

{
"ImageUrl": "https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg",
"Filter": "value > 10",
"MatchThreshold": "1",
"Limit": "30",
"Offset": "0",
"GroupId": "work"
}
```

输出示例

```
{
"Response": {
"Count": 1,
"ImageInfos": [
{
"CustomContent": "",
"EntityId": "work-1",
"PicName": "work-1-1",
"Score": 100,
"Tags": "{\"my_tag\": \"1\"}"
}
],
"Object": {
"AllBox": [
{
"CategoryId": 0,
"Rect": {
"Height": 268,
```

```
"Width": 193,
"X": 278,
"Y": 162
},
"Score": 60
},
{
"CategoryId": 4,
"Rect": {
"Height": 138,
"Width": 132,
"X": 337,
"Y": 1053
},
"Score": 78
},
{
"CategoryId": 5,
"Rect": {
"Height": 31,
"Width": 83,
"X": 356,
"Y": 135
},
"Score": 27
}
],
"Attributes": [
{
"Details": "渐层",
"Type": "图案"
},
{
"Details": "街头",
"Type": "风格"
},
{
"Details": "灰色",
"Type": "颜色"
},
{
"Details": "尼龙",
"Type": "材质"
},
{
"Details": "连帽",
"Type": "颈线设计"
},
{
"Details": "男装",
"Type": "类型"
```

```
},
{
"Details": "加长款",
"Type": "衣长"
},
{
"Details": "连衣裤",
"Type": "类别"
}
],
"Box": {
"CategoryId": 0,
"Rect": {
"Height": 268,
"Width": 193,
"X": 278,
"Y": 162
},
"Score": 60
},
"CategoryId": 0,
"Colors": [
{
"Color": "293133",
"Label": "Black-darkgrey",
"Percentage": 10
},
{
"Color": "4A3D46",
"Label": "Black-grapethistle",
"Percentage": 8
},
{
"Color": "808080",
"Label": "Gray-gray",
"Percentage": 7
},
{
"Color": "555555",
"Label": "Blue-ebony",
"Percentage": 6
}
]
},
"RequestId": "7ddbbc94-1adc-45f7-a114-8fdebaabf6d4"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.BalanceInsufficient	余额不足，开通失败，请充值后再开通。
FailedOperation.DownloadError	文件下载错误。
FailedOperation.EmptyImageError	图片内容为空。
FailedOperation.ImageDecodeFailed	图片解码失败。
FailedOperation.ImageGroupChargeStatusClose	停止服务，控制台开关关闭。
FailedOperation.ImageGroupEmpty	图库为空。
FailedOperation.ImageNotSupported	不支持的图片文件。
FailedOperation.ImageResolutionExceed	图片分辨率过大。
FailedOperation.ImageResolutionInsufficient	图片分辨率过小。
FailedOperation.ImageSearchInvalid	未查询到结果。
FailedOperation.ImageSizeExceed	base64编码后的图片数据过大。
FailedOperation.ImageUrlInvalid	url地址不合法，无法下载。
FailedOperation.InnerError	内部错误。
FailedOperation.RequestError	后端服务请求失败。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
FailedOperation.ServerError	算法服务异常，请重试。
FailedOperation.UnKnowError	内部错误。

错误码	描述
FailedOperation.Unknown	未知错误。
InvalidParameter.ImageFormatNotSupport	不支持的图片格式。
InvalidParameter.InvalidParameter	参数取值错误。
InvalidParameter.PictureSolidColorError	图片不可为纯色图。
InvalidParameterValue.EntityIdTooLong	物品ID超出长度限制。
InvalidParameterValue.FilterInvalid	Filter参数不合法。
InvalidParameterValue.FilterSizeExceed	Filter参数过长。
InvalidParameterValue.ImageGroupIdIllegal	图库ID不合法。
InvalidParameterValue.ImageGroupIdNotExist	图库ID不存在。
InvalidParameterValue.ImageGroupIdTooLong	图库ID超出长度限制。
InvalidParameterValue.LimitExceed	返回数量不在合法范围内。
MissingParameter.ErrorParameterEmpty	必选参数为空。
RequestLimitExceeded	请求的次数超过了频率限制。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.IsOpening	服务正在开通中，请稍等。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。

更新图片

最近更新时间：2025-06-19 16:42:51

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

本接口支持根据图库ID、物品ID、图片名称来修改图片信息（暂仅支持修改Tags）

- 可前往 图像搜索 产品文档中查看更多产品信息。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：UpdateImage。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
GroupId	是	String	图库ID。 示例值：test_group
EntityId	是	String	物品ID，最多支持64个字符。 示例值：test_entity
PicName	否	String	图片名称，最多支持64个字符。 示例值：my_pic
Tags	否	String	新的自定义标签，最多不超过10个，格式为JSON。 示例值：{"flag":"1"}

3. 输出参数

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

4. 示例

示例1 更新图片接口调用失败示例

输入示例

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

{
"GroupId": "my_group",
"EntityId": "entity_297",
"PicName": "my_pic"
}
```

输出示例

```
{
"Response": {
"Error": {
"Code": "InvalidParameter.InvalidParameter",
"Message": "参数不合法。"
},
"RequestId": "cac925c7-c939-4eb6-8db5-d64123385c30"
}
}
```

示例2 更新图片接口调用成功示例

输入示例

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

{
"GroupId": "group_972",
"EntityId": "entity_297",
"PicName": "my_pic"
}
```

输出示例

```
{
"Response": {
"RequestId": "cac925c7-c939-4e26-8db5-d642f7385c30"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.DownloadError	文件下载错误。
FailedOperation.ImageDecodeFailed	图片解码失败。
FailedOperation.ImageGroupEmpty	图库为空。
FailedOperation.ImageNotFoundInfo	未找到图片信息。
FailedOperation.ImageNotSupported	不支持的图片文件。
FailedOperation.ImageSizeExceed	base64编码后的图片数据过大。
FailedOperation.ImageUrlInvalid	url地址不合法，无法下载。
FailedOperation.RequestError	后端服务请求失败。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
FailedOperation.ServerError	算法服务异常，请重试。
InvalidParameter.ImageFormatNotSupport	不支持的图片格式。
InvalidParameter.InvalidParameter	参数取值错误。
InvalidParameterValue.EntityIdEmpty	物品ID为空。
InvalidParameterValue.ImageGroupIdIllegal	图库ID不合法。
InvalidParameterValue.ImageGroupIdNotExist	图库ID不存在。

错误码	描述
InvalidParameterValue.ImageGroupIdTooLong	图库ID超出长度限制。
InvalidParameterValue.PicNameEmpty	图片名称为空。
InvalidParameterValue.PicNameTooLong	图片名称超出长度限制。
InvalidParameterValue.TagsKeysExceed	标签数量过多。
InvalidParameterValue.TagsValueIllegal	标签值类型不合法。
InvalidParameterValue.TagsValueSizeExceed	标签值长度过长。
MissingParameter.ErrorParameterEmpty	必选参数为空。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。

图像识别相关接口
商品识别

最近更新时间：2025-06-19 16:42:50

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

本接口支持识别图片中包含的商品，能够输出商品的品类名称、类别，还可以输出商品在图片中的位置。支持一张图片多个商品的识别。

> 说明：
>
> - 公共参数中的签名方式必须指定为V3版本，即配置SignatureMethod参数为TC3-HMAC-SHA256。

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：DetectProduct。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
ImageUrl	否	String	图片URL地址。 图片限制： • 图片格式：PNG、JPG、JPEG。 • 图片大小：所下载图片经Base64编码后不超过4M。图片下载时间不超过3秒。 建议： • 图片像素：大于50*50像素，否则影响识别效果； • 长宽比：长边：短边<5； 接口响应时间会受到图片下载时间的影响，建议使用更可靠的存储服务，推荐将图片存储在腾讯云COS。 示例值：https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg
ImageBase64	否	String	图片经过base64编码的内容。最大不超过4M。与ImageUrl同时存在时优先使用ImageUrl字段。 **注意：图片需要base64编码，并且要去掉编码头部。** 示例值：/9j/4AAQSkZJRgABAQAAAQABAAD/4glo...lftXF/DjFZNXoSP5V2U0HMt/1FQf/Z

3. 输出参数

参数名称	类型	描述
Products	Array of Product	商品识别结果数组 示例值：[{"Name":"男士西服套装","Parents":"服饰内衣-男装","Confidence":59,"XMin":336,"YMin":191,"XMax":799,"YMax":775},{"Name":"休闲鞋","Parents":"鞋靴-时尚女鞋","Confidence":40,"XMin":466,"YMin":1209,"XMax":695,"YMax":1377}]

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

4. 示例

示例1 商品识别检测请求成功

输入示例

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

{
"ImageUrl": "http://a.vpimg2.com/upload/merchandise/41498/CABBEEN-3030701301-1.jpg"
}
```

输出示例

```
{
"Response": {
"Products": [
{
"Name": "男士西服套装",
"Parents": "服饰内衣-男装",
"Confidence": 59,
"XMin": 336,
"YMin": 191,
"XMax": 799,
"YMax": 775
},
{
"Name": "休闲鞋",
"Parents": "鞋靴-时尚女鞋",
"Confidence": 40,
"XMin": 466,
"YMin": 1209,
"XMax": 695,
"YMax": 1377
}
],
"RequestId": "2bd4243f-4d26-4246-a5f4-0f2dbc730d62"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.BalanceInsufficient	余额不足，开通失败，请充值后再开通。
FailedOperation.DownloadError	文件下载错误。
FailedOperation.ImageDecodeFailed	图片解码失败。
FailedOperation.ImageNotSupported	不支持的图片文件。
FailedOperation.InvokeChargeError	调用计费返回失败。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
FailedOperation.TooLargeFileError	文件太大。
FailedOperation.UnKnowError	内部错误。
FailedOperation.UnOpenError	服务未开通。
InvalidParameter.InvalidParameter	参数取值错误。
InvalidParameterValue.InvalidParameterValueLimit	参数值错误。
LimitExceeded.TooLargeFileError	文件太大。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。
ResourcesSoldOut.ChargeStatusException	计费状态异常。

车辆识别（增强版）

最近更新时间：2025-06-19 16:42:49

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

车辆识别（增强版）在车辆识别的基础上**增加了车牌识别的功能，并升级了车型识别的效果**。可对图片中车辆的车型和车牌进行同时识别，输出车辆的车牌信息，以及车辆品牌（如路虎）、车系（如神行者2）、类型（如中型SUV）、颜色和车辆在图中的坐标等信息，覆盖轿车、SUV、大型客车等市面常见车，支持三千多种车辆型号。如果图片中存在多辆车，会分别输出每辆车的车型、车牌和坐标。

> 说明：
>
> - 公共参数中的签名方式必须指定为V3版本，即配置SignatureMethod参数为TC3-HMAC-SHA256。

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：RecognizeCarPro。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
ImageUrl	否	String	图片URL地址。 图片限制： • 图片格式：PNG、JPG、JPEG。 • 图片大小：所下载图片经Base64编码后不超过4M。图片下载时间不超过3秒。 建议： • 图片像素：大于50*50像素，否则影响识别效果； • 长宽比：长边：短边<5； 接口响应时间会受到图片下载时间的影响，建议使用更可靠的存储服务，推荐将图片存储在腾讯云COS。 示例值：https://cos.ap-singapore.myqcloud.com/input.png
ImageBase64	否	String	图片经过base64编码的内容。最大不超过4M。与ImageUrl同时存在时优先使用ImageUrl字段。 **注意：图片需要base64编码，并且要去掉编码头部。** 支持的图片格式：PNG、JPG、JPEG、BMP，暂不支持GIF格式。支持的图片大小：所下载图片经Base64编码后不超过4M。图片下载时间不超过3秒。 示例值：/9j/4AAQSkZJRgABAQAAAQABAAD/4glo...lftXF/DjFZNXoSP5V2U0HMt/1FQf/Z

3. 输出参数

参数名称	类型	描述
CarCoords	Array of Coord	汽车的四个矩形顶点坐标，如果图片中存在多辆车，则输出最大车辆的坐标。 示例值：[{"X":87,"Y":291},{"X":87,"Y":1},{"X":540,"Y":1},{"X":540,"Y":291}]

参数名称	类型	描述
CarTags	Array of CarTagItem	车辆属性识别的结果数组，如果识别到多辆车，则会输出每辆车的top1结果。 注意：置信度是指车牌信息置信度。
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 调用失败示例

输入图片URL，未检测到车辆信息。

输入示例

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

{
"ImageUrl": "https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg"
}
```

输出示例

```
{
"Response": {
"Error": {
"Code": "FailedOperation.NoObjectDetected",
"Message": "未检测到目标。"
},
"RequestId": "ad418ac5-fbfd-4bd7-8f4a-35ab085e27dd"
}
}
```

示例2 车辆识别示例代码

输入图片URL，成功检测到车辆信息。

输入示例

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

{
```

```
"ImageUrl": "https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg"
}
```

输出示例


```
{
"Response": {
"CarCoords": [
{
"X": 87,
"Y": 291
},
{
"X": 87,
"Y": 1
},
{
"X": 540,
"Y": 1
},
{
"X": 540,
"Y": 291
}
],
"CarTags": [
{
"Brand": "奔驰",
"CarLocation": [
{
"X": 87,
"Y": 291
},
{
"X": 87,
"Y": 1
},
{
"X": 540,
"Y": 1
},
{
"X": 540,
"Y": 291
}
],
"Color": "绿",
"ColorConfidence": 67,
"Confidence": 64,
"Orientation": "车头",
"OrientationConfidence": 99,
```

```
"PlateConfidence": 99,
"PlateContent": {
"Color": "蓝色",
"Plate": "浙G29XY9",
"PlateAngle": 13.560508,
"PlateLocation": [
{
"X": 452,
"Y": 227
},
{
"X": 452,
"Y": 190
},
{
"X": 507,
"Y": 190
},
{
"X": 507,
"Y": 227
}
],
"PlateStatus": "正常车牌",
"PlateStatusConfidence": 99,
"Type": "普通蓝牌"
},
"Serial": "奔驰GLC级",
"Type": "SUV",
"TypeConfidence": 54,
"Year": 0
}
],
"RequestId": "2aba0901-31b9-456f-ad38-a2e7075d0965"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.BalanceInsufficient	余额不足，开通失败，请充值后再开通。
FailedOperation.DownloadError	文件下载错误。
FailedOperation.EmptyImageError	图片内容为空。
FailedOperation.ImageDecodeFailed	图片解码失败。
FailedOperation.ImageNotSupported	不支持的图片文件。
FailedOperation.ImageSizeExceed	base64编码后的图片数据过大。
FailedOperation.ImageUnQualified	图片不满足检测要求。
FailedOperation.RecognizeFailded	车辆识别失败。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
FailedOperation.UnKnowError	内部错误。
FailedOperation.UnOpenError	服务未开通。
FailedOperation.Unknown	未知错误。
InvalidParameter.InvalidParameter	参数取值错误。
InvalidParameterValue.InvalidParameterValueLimit	参数值错误。
InvalidParameterValue.UrlIllegal	URL格式不合法。
LimitExceeded.TooLargeFileError	文件太大。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。
ResourcesSoldOut.ChargeStatusException	计费状态异常。

车辆识别

最近更新时间：2025-06-19 16:42:49

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

车辆识别可对图片中车辆的车型进行识别，输出车辆的品牌（如路虎）、车系（如神行者2）、类型（如中型SUV）、颜色和车辆在图中的坐标等信息，覆盖轿车、SUV、大型客车等市面常见车，支持三千多种车辆型号。如果图片中存在多辆车，会分别输出每辆车的车型和坐标。

> 说明：
>
> - 公共参数中的签名方式必须指定为V3版本，即配置SignatureMethod参数为TC3-HMAC-SHA256。

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：RecognizeCar。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
ImageUrl	否	String	图片URL地址。 图片限制： • 图片格式：PNG、JPG、JPEG。 • 图片大小：所下载图片经Base64编码后不超过4M。图片下载时间不超过3秒。 建议： • 图片像素：大于50*50像素，否则影响识别效果； • 长宽比：长边：短边<5； 接口响应时间会受到图片下载时间的影响，建议使用更可靠的存储服务，推荐将图片存储在腾讯云COS。 示例值：https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg
ImageBase64	否	String	图片经过base64编码的内容。最大不超过4M。与ImageUrl同时存在时优先使用ImageUrl字段。 **注意：图片需要base64编码，并且要去掉编码头部。** 支持的图片格式：PNG、JPG、JPEG、BMP，暂不支持GIF格式。支持的图片大小：所下载图片经Base64编码后不超过4M。图片下载时间不超过3秒。 示例值：/9j/4AAQSkZJRgABAQAAAQABAAD/4gIo...lftXF/DjFZNXoSP5V2U0HMt/1FQf/Z

3. 输出参数

参数名称	类型	描述
CarCoords	Array of Coord	汽车的四个矩形顶点坐标，如果图片中存在多辆车，则输出最大车辆的坐标。 示例值：[{"X":513,"Y":651},{"X":513,"Y":441},{"X":780,"Y":441},{"X":780,"Y":651}]

参数名称	类型	描述
CarTags	Array of CarTagItem	车辆属性识别的结果数组，如果识别到多辆车，则会输出每辆车的top1结果。 示例值：[{"Serial":"途安","Brand":"大众","Type":"MPV","Color":"白","Confidence":93,"Year":2008,"CarLocation":[{"X":513,"Y":651},{"X":513,"Y":441},{"X":780,"Y":441},{"X":780,"Y":651}]
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 车辆识别示例代码

输入示例

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

{
"ImageUrl": "https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg"
}
```

输出示例

```
{
"Response": {
"CarCoords": [
{
"X": 513,
"Y": 651
},
{
"X": 513,
"Y": 441
},
{
"X": 780,
"Y": 441
},
{
"X": 780,
"Y": 651
}
],
"CarTags": [
{
"Serial": "途安",
"Brand": "大众",
```

```
"Type": "MPV",
"Color": "白",
"Confidence": 93,
"Year": 2008,
"CarLocation": [
{
"X": 513,
"Y": 651
},
{
"X": 513,
"Y": 441
},
{
"X": 780,
"Y": 441
},
{
"X": 780,
"Y": 651
}
],
"PlateContent": null,
"PlateConfidence": null,
"TypeConfidence": null,
"ColorConfidence": null,
"Orientation": null,
"OrientationConfidence": null
},
{
"Serial": "君越",
"Brand": "别克",
"Type": "中型车",
"Color": "银灰",
"Confidence": 47,
"Year": 2009,
"CarLocation": [
{
"X": 45,
"Y": 658
},
{
"X": 45,
"Y": 459
},
{
"X": 313,
"Y": 459
},
{
"X": 313,
```

```
"Y": 658
}
],
"PlateContent": null,
"PlateConfidence": null,
"TypeConfidence": null,
"ColorConfidence": null,
"Orientation": null,
"OrientationConfidence": null
}
],
"RequestId": "81235fb9-d91e-4a40-91ac-9ef5675954a2"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.DownloadError	文件下载错误。
FailedOperation.EmptyImageError	图片内容为空。
FailedOperation.ImageDecodeFailed	图片解码失败。
FailedOperation.ImageNotSupported	不支持的图片文件。
FailedOperation.ImageSizeExceed	base64编码后的图片数据过大。

错误码	描述
FailedOperation.ImageUnQualified	图片不满足检测要求。
FailedOperation.RecognizeFailded	车辆识别失败。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
FailedOperation.UnKnowError	内部错误。
FailedOperation.UnOpenError	服务未开通。
FailedOperation.Unknown	未知错误。
InvalidParameter.InvalidParameter	参数取值错误。
InvalidParameterValue.InvalidParameterValueLimit	参数值错误。
InvalidParameterValue.UrlIllegal	URL格式不合法。
LimitExceeded.TooLargeFileError	文件太大。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。
ResourcesSoldOut.ChargeStatusException	计费状态异常。

文件封识别

最近更新时间：2025-06-19 16:42:50

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

文件封识别可检测图片中是否包含符合文件封（即文件、单据、资料等的袋状包装）特征的物品，覆盖顺丰快递文件封、文件袋、档案袋等多种文件封类型，可应用于物流行业对文件快递的包装审核等场景。

> 说明：
>
> - 公共参数中的签名方式必须指定为V3版本，即配置SignatureMethod参数为TC3-HMAC-SHA256。

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：DetectEnvelope。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
ImageUrl	否	String	图片的URL地址。图片存储于腾讯云的Url可保障更高下载速度和稳定性，建议图片存储于腾讯云。 非腾讯云存储的Url速度和稳定性可能受一定影响。 图片大小的限制为4M，图片像素的限制为4k。 示例值：https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg
ImageBase64	否	String	图片经过base64编码的内容。与ImageUrl同时存在时优先使用ImageUrl字段。 图片大小的限制为4M，图片像素的限制为4k。 **注意：图片需要base64编码，并且要去掉编码头部。 示例值：/9j/4AAQSkZJRgABAQAAAQABAAD/4glo...lftXF/DjFZNXoSP5V2U0HMt/1FQf/Z

3. 输出参数

参数名称	类型	描述
FirstTags	Array of ImageTag	一级标签结果数组。识别是否文件封。 示例值：[{"Name":"非文件封","Confidence":97.76},{"Name":"文件封","Confidence":2.24}]
SecondTags	Array of ImageTag	二级标签结果数组。识别文件封正反面。 示例值：[{"Name":"非文件封","Confidence":97.5},{"Name":"非顺丰文件封正面","Confidence":2.28},{"Name":"顺丰文件封正面","Confidence":0.09},{"Name":"非顺丰文件封反面","Confidence":0.07},{"Name":"顺丰文件封反面","Confidence":0.05}]
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

参数名称	类型	描述
		RequestId ）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 调用失败示例

输入示例

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

{
"ImageUrl": "https://cos.ap-singapore.myqcloud.com/input.png"
}
```

输出示例

```
{
"Response": {
"Error": {
"Code": "FailedOperation.DownloadError ",
"Message": "文件下载错误。"
},
"RequestId": "ad418ac5-fbfd-4bd7-8f4a-35ab085e27dd"
}
}
```

示例2 调用成功示例

输入示例

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

{
"ImageUrl": "https://cos.ap-singapore.myqcloud.com/input.png"
}
```

输出示例

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

```
"FirstTags": [
{
"Name": "非文件封",
"Confidence": 97.76
},
{
"Name": "文件封",
"Confidence": 2.24
}
],
"SecondTags": [
{
"Name": "非文件封",
"Confidence": 97.5
},
{
"Name": "非顺丰文件封正面",
"Confidence": 2.28
},
{
"Name": "顺丰文件封正面",
"Confidence": 0.09
},
{
"Name": "非顺丰文件封反面",
"Confidence": 0.07
},
{
"Name": "顺丰文件封反面",
"Confidence": 0.05
}
],
"RequestId": "c7552894-0e1c-468d-afc7-685483c3e3be"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.DownloadError	文件下载错误。
FailedOperation.ImageDecodeFailed	图片解码失败。
FailedOperation.ImageNotSupported	不支持的图片文件。
FailedOperation.ImageSizeExceed	base64编码后的图片数据过大。
FailedOperation.ImageUrlInvalid	url地址不合法，无法下载。
FailedOperation.InnerError	内部错误。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
FailedOperation.TooLargeFileError	文件太大。
InvalidParameter.InvalidParameter	参数取值错误。
InvalidParameterValue.UrlIllegal	URL格式不合法。
LimitExceeded.TooLargeFileError	文件太大。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。

厨师穿戴识别接口

最近更新时间：2025-06-19 16:42:50

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

可对图片中厨师穿戴进行识别，支持厨师服识别，厨师帽识别，赤膊识别和口罩识别,可应用于明厨亮灶场景。
"被优选过滤"标签值在人体优选开关开启时才会返回。
厨师服：厨师服定义为白色上衣
厨师服识别(酒店版)：厨师服定义为红色，白色，黑色上衣

序号	标签名称	标签值
1	厨师服识别	无厨师服、有厨师服、被优选过滤
2	厨师服识别（酒店版）	无厨师服、有厨师服、被优选过滤
3	厨师帽识别	无厨师帽、有厨师帽、被优选过滤
4	赤膊识别	非赤膊、赤膊、被优选过滤
5	口罩识别	无口罩、有口罩、口罩不确定、被优选过滤

默认接口请求频率限制：2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：DetectChefDress。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
ImageUrl	否	String	图片的 Url 。 ImageUrl和ImageBase64必须提供一个，同时存在时优先使用ImageUrl字段。 图片限制： • 图片格式：支持PNG、JPG、JPEG、不支持 GIF 图片。 • 图片大小：对应图片 base64 编码后大小不可超过5M。图片分辨率不超过 3840 x 2160pixel。 建议： • 接口响应时间会受到图片下载时间的影响，建议使用更可靠的存储服务，推荐将图片存储在腾讯云 COS。 示例值：https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg
ImageBase64	否	String	图片经过base64编码的内容。与ImageUrl同时存在时优先使用ImageUrl字段。 注意：图片需要base64编码，并且要去掉编码头部。 支持的图片格式：PNG、JPG、JPEG、暂不支持GIF格式。 支持的图片大小：所下载图片经Base64编码后不超过5M。

参数名称	必选	类型	描述
			示例 值：/9j/4AAQSkZJRgABAQAAAQABAAD/4glo...lftXF/DjFZNXoSP5V2U0HMt/1FQf/Z
EnableDetect	否	Boolean	人体检测模型开关，“true”为开启，“false”为关闭 默认为开启，开启后可先对图片中的人体进行检测之后再进行属性识别 示例值：true
EnablePreferred	否	Boolean	人体优选开关，“true”为开启，“false”为关闭 开启后自动对检测质量低的人体进行优选过滤，有助于提高属性识别的准确率。 默认为开启，仅在人体检测开关开启时可配置，人体检测模型关闭时人体优选也关闭 人体优选开启时，检测到的人体分辨率不超过1920*1080 pixel 示例值：true

3. 输出参数

参数名称	类型	描述
Bodies	Array of AttributesForBody	识别到的人体属性信息。单个人体属性信息包括人体检测置信度，属性信息，人体检测框。
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 厨师穿戴识别请求失败

输入示例

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

{
"ImageUrl": "https://123.jpg"
}
```

输出示例

```
{
"Response": {
"Error": {
"Code": "FailedOperation.DownLoadError",
"Message": "下载失败"
},
"RequestId": "a169390a-6ff3-4c42-ad25-a7858c35e576"
}
}
```

示例2 厨师穿戴识别请求成功

输入示例

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

{
"ImageUrl": "https://test.jpg"
}
```

输出示例

```
{
"Response": {
"Bodies": [
{
"Attributes": [
{
"Confidence": 0.6875998,
"Label": "无厨师服",
"Name": "厨师服识别"
},
{
"Confidence": 0.6875998,
"Label": "无厨师服",
"Name": "厨师服识别（酒店版）"
},
{
"Confidence": 0.9608861,
"Label": "无厨师帽",
"Name": "厨师帽识别"
},
{
"Confidence": 0,
"Label": "被优选过滤",
"Name": "口罩识别"
},
{
"Confidence": 0.99437755,
"Label": "赤膊",
"Name": "赤膊识别"
}
],
"DetectConfidence": 0.914,
"Rect": {
"Height": 1110,
"Width": 440,
"X": 217,
"Y": 79
```

```
}
}
],
"RequestId": "a9050291-41a7-48ca-bd94-7a2a25d1d466"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.DownLoadError	文件下载失败。
FailedOperation.DownloadError	文件下载错误。
FailedOperation.ImageDecodeFailed	图片解码失败。
FailedOperation.ImageResolutionExceed	图片分辨率过大。
FailedOperation.ImageSizeExceed	base64编码后的图片数据过大。
FailedOperation.InnerError	内部错误。
FailedOperation.NoBodyInPhoto	图片中没有人体。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
FailedOperation.TooLargeFileError	文件太大。
FailedOperation.Unknown	未知错误。

错误码	描述
InvalidParameter.InvalidParameter	参数取值错误。
InvalidParameterValue.ImageEmpty	图片为空。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。

宠物识别

最近更新时间：2025-06-19 16:42:50

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

传入一张图片，识别出图片中是否存在宠物

- 公共参数中的签名方式必须指定为V3版本，即配置SignatureMethod参数为TC3-HMAC-SHA256。

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：DetectPet。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
ImageUrl	否	String	图片的URL地址。图片存储于腾讯云的Url可保障更高下载速度和稳定性，建议图片存储于腾讯云。 非腾讯云存储的Url速度和稳定性可能受一定影响。 图片大小的限制为4M，图片像素的限制为4k。 示例值：https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg
ImageBase64	否	String	图片经过base64编码的内容。与ImageUrl同时存在时优先使用ImageUrl字段。 图片大小的限制为4M，图片像素的限制为4k。 **注意：图片需要base64编码，并且要去掉编码头部。** 示例值：/9j/4AAQSkZJRgABAQAAAQABAAD/4glo...lftXF/DjFZNXoSP5V2U0HMt/1FQf/Z

3. 输出参数

参数名称	类型	描述
Pets	Array of Pet	识别出图片中的宠物信息列表。 示例值：[{"Score":99,"Name":"cat","Location":{"Y":121,"X":36,"Height":203,"Width":334}}]
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 识别失败

输入示例

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

{
"ImageUrl": "https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg"
}
```

输出示例

```
{
"Response": {
"Error": {
"Code": "FailedOperation.DownLoadError",
"Message": "下载失败"
},
"RequestId": "a169390a-6ff3-4c42-ad25-a7858c35e576"
}
}
```

示例2 识别成功

输入示例

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

{
"ImageUrl": "https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg"
}
```

输出示例

```
{
"Response": {
"RequestId": "a169390a-6ff3-4c42-ad25-a7858c35e576",
"Pets": [
{
"Score": 99,
"Name": "cat",
"Location": {
"Y": 121,
"X": 36,
```

```
"Height": 203,
"Width": 334
}
}
]
}
}
```

示例3 图片中未识别出宠物

输入示例

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

{
"ImageUrl": "https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg"
}
```

输出示例

```
{
"Response": {
"Error": {
"Code": "FailedOperation.NoObjectDetected",
"Message": "未检测到目标。"
},
"RequestId": "ad418ac5-fbfd-4bd7-8f4a-35ab085e27dd"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.BalanceInsufficient	余额不足，开通失败，请充值后再开通。
FailedOperation.DownloadError	文件下载错误。
FailedOperation.ImageDecodeFailed	图片解码失败。
FailedOperation.ImageNotSupported	不支持的图片文件。
FailedOperation.ImageSizeExceed	base64编码后的图片数据过大。
FailedOperation.InnerError	内部错误。
FailedOperation.InvokeChargeError	调用计费返回失败。
FailedOperation.NoObjectDetected	未检测到目标。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
FailedOperation.UnKnowError	内部错误。
FailedOperation.Unknown	未知错误。
InvalidParameter.InvalidParameter	参数取值错误。
InvalidParameterValue.InvalidParameterValueLimit	参数值错误。
InvalidParameterValue.UrlIllegal	URL格式不合法。
LimitExceeded.TooLargeFileError	文件太大。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。
ResourcesSoldOut.ChargeStatusException	计费状态异常。

安全属性识别

最近更新时间：2025-06-19 16:42:49

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

安全属性识别可对图片中人体安全防护属性进行识别，支持识别安全帽，反光衣，护目镜，工服，手套，工地安全带，口罩，抽烟，玩手机等多种属性。"被优选过滤"标签值在人体优选开关开启时才会返回。

序号	标签名称	标签值
1	安全帽识别	无安全帽、有安全帽、被优选过滤
2	玩手机识别	没有电话、打电话、玩手机、被优选过滤
3	抽烟识别	没有抽烟、抽烟、被优选过滤
4	口罩识别	无口罩、有口罩、口罩不确定、被优选过滤
5	工地安全带识别	无工地安全带、工地安全带、被优选过滤
6	手套识别	无手套、有手套、手套不确定、被优选过滤
7	工服识别	无工服、有工服、被优选过滤
8	护目镜识别	无护目镜、有护目镜、被优选过滤
9	反光衣识别	无反光衣、有反光衣、被优选过滤

默认接口请求频率限制：2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：DetectSecurity。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
ImageUrl	否	String	图片的 Url 。 ImageUrl和ImageBase64必须提供一个，同时存在时优先使用ImageUrl字段。 图片限制： • 图片格式：支持PNG、JPG、JPEG、不支持 GIF 图片。 • 图片大小：对应图片 base64 编码后大小不可超过5M。图片分辨率不超过3840 x 2160 pixel。 建议： • 接口响应时间会受到图片下载时间的影响，建议使用更可靠的存储服务，推荐将图片存储在腾讯云 COS。 示例值：https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg

参数名称	必选	类型	描述
ImageBase64	否	String	图片经过base64编码的内容。 最大不超过4M。与ImageUrl同时存在时优先使用ImageUrl字段。 注意：图片需要base64编码，并且要去掉编码头部。 支持的图片格式：PNG、JPG、JPEG、暂不支持GIF格式。 支持的图片大小：所下载图片经Base64编码后不超过5M。 示例 值：/9j/4AAQSkZJRgABAQAAAQABAAD/4glo...lftXF/DjFZNXoSP5V2U0HMt/1FQf/Z
EnableDetect	否	Boolean	人体检测模型开关，“true”为开启，“false”为关闭 开启后可先对图片中的人体进行检测之后再进行属性识别，默认为开启 示例值：true
EnablePreferred	否	Boolean	人体优选开关，“true”为开启，“false”为关闭 开启后自动对检测质量低的人体进行优选过滤，有助于提高属性识别的准确率。 默认为开启，仅在人体检测开关开启时可配置，人体检测模型关闭时人体优选也关闭 如开启人体优选，检测到的人体分辨率需不大于1920*1080 pixel 示例值：ture

3. 输出参数

参数名称	类型	描述
Bodies	Array of AttributesForBody	识别到的人体属性信息。单个人体属性信息包括人体检测置信度，属性信息，人体检测框。
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 安全属性识别请求失败

安全属性识别请求失败

输入示例

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

{
"ImageUrl": "https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg"
}
```

输出示例

```
{
"Response": {
"Error": {
"Code": "FailedOperation.DownLoadError",
"Message": "下载失败"
},
```

```
"RequestId": "a169390a-6ff3-4c42-ad25-a7858c35e516"
}
}
```

示例2 安全属性识别请求成功

安全属性识别请求成功

输入示例

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

{
"ImageUrl": "https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg"
}
```

输出示例

```
{
"Response": {
"Bodies": [
{
"Attributes": [
{
"Confidence": 0.9608861,
"Label": "无安全帽",
"Name": "安全帽识别"
},
{
"Confidence": 0.9768763,
"Label": "没有电话",
"Name": "玩手机识别"
},
{
"Confidence": 0.8206851,
"Label": "没有抽烟",
"Name": "抽烟识别"
},
{
"Confidence": 0,
"Label": "被优选过滤",
"Name": "口罩识别"
},
{
"Confidence": 0.9113377,
"Label": "无工地安全带",
"Name": "工地安全带识别"
```

```
},
{
"Confidence": 0.7201198,
"Label": "无手套",
"Name": "手套识别"
},
{
"Confidence": 0.6875998,
"Label": "无工服",
"Name": "工服识别"
},
{
"Confidence": 0.611658,
"Label": "无护目镜",
"Name": "护目镜识别"
},
{
"Confidence": 0.9828068,
"Label": "无反光衣",
"Name": "反光衣识别"
}
],
"DetectConfidence": 0.914,
"Rect": {
"Height": 1110,
"Width": 440,
"X": 217,
"Y": 79
}
}
],
"RequestId": "8a3f48ea-f995-4ad2-9ff5-7f8faf8bdbaf"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.DownLoadError	文件下载失败。
FailedOperation.DownloadError	文件下载错误。
FailedOperation.ImageDecodeFailed	图片解码失败。
FailedOperation.ImageResolutionExceed	图片分辨率过大。
FailedOperation.ImageSizeExceed	base64编码后的图片数据过大。
FailedOperation.InnerError	内部错误。
FailedOperation.NoBodyInPhoto	图片中没有人体。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
FailedOperation.TooLargeFileError	文件太大。
FailedOperation.UnKnowError	内部错误。
InvalidParameter.InvalidParameter	参数取值错误。
InvalidParameterValue.ImageEmpty	图片为空。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。

图像处理相关接口
图像质量评估

最近更新时间：2025-06-19 16:42:54

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

评估输入图片在视觉上的质量，从多个方面评估，并同时给出综合的、客观的清晰度评分，和主观的美观度评分。

- 可前往 图像处理 产品文档中查看更多产品信息。
- 公共参数中的签名方式必须指定为V3版本，即配置SignatureMethod参数为TC3-HMAC-SHA256。

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：AssessQuality。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
ImageUrl	否	String	图片URL地址。 图片限制： • 图片格式：PNG、JPG、JPEG。 • 图片大小：所下载图片经Base64编码后不超过4M。图片下载时间不超过3秒。 建议： • 图片像素：大于50*50像素，否则影响识别效果。 • 长宽比：长边：短边<5。 接口响应时间会受到图片下载时间的影响，建议使用更可靠的存储服务，推荐将图片存储在腾讯云COS。 示例值：https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg
ImageBase64	否	String	图片经过Base64编码的内容。最大不超过4M。与ImageUrl同时存在时优先使用ImageUrl字段。 **注意：图片需要Base64编码，并且要去掉编码头部。** 示例值：/9j/4AAQSkZJRgABAQAAAQABAAD/4glo...lftXF/DjFZNXoSP5V2U0HMt/1FQf/Z

3. 输出参数

参数名称	类型	描述
LongImage	Boolean	取值为TRUE或FALSE，TRUE为长图，FALSE为正常图，长图定义为长宽比大于等于3或小于等于1/3的图片。 示例值：true

参数名称	类型	描述
BlackAndWhite	Boolean	取值为TRUE或FALSE，TRUE为黑白图，FALSE为否。黑白图即灰度图，指红绿蓝三个通道都是以灰度色阶显示的图片，并非视觉上的“黑白图片”。 示例值：true
SmallImage	Boolean	取值为TRUE或FALSE，TRUE为小图，FALSE为否, 小图定义为最长边小于179像素的图片。当一张图片被判断为小图时，不建议做推荐和展示，其他字段统一输出为0或FALSE。 示例值：true
BigImage	Boolean	取值为TRUE或FALSE，TRUE为大图，FALSE为否，定义为最短边大于1000像素的图片 示例值：1
PureImage	Boolean	取值为TRUE或FALSE，TRUE为纯色图或纯文字图，即没有内容或只有简单内容的图片，FALSE为正常图片。 示例值：true
ClarityScore	Integer	综合评分。图像清晰度的得分，对图片的噪声、曝光、模糊、压缩等因素进行综合评估，取值为[0, 100]，值越大，越清晰。一般大于50为较清晰图片，标准可以自行把握。 示例值：93
AestheticScore	Integer	综合评分。图像美观度得分， 从构图、色彩等多个艺术性维度评价图片，取值为[0, 100]，值越大，越美观。一般大于50为较美观图片，标准可以自行把握。 示例值：92
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 图像质量评估请求成功

输入示例

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

{
"ImageUrl": "https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg"
}
```

输出示例

```
{
"Response": {
"LongImage": false,
"BlackAndWhite": false,
"SmallImage": false,
"BigImage": true,
"PureImage": false,
"ClarityScore": 93,
"AestheticScore": 92,
"RequestId": "bfd478e1-5c94-4e37-ad22-4a5224a09492"
```

```
}
}
```

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.InvalidAuthorization	认证失败。
FailedOperation.DownloadError	文件下载错误。
FailedOperation.ImageDecodeFailed	图片解码失败。
FailedOperation.ImageNotSupported	不支持的图片文件。
FailedOperation.InvokeChargeError	调用计费返回失败。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
FailedOperation.UnKnowError	内部错误。
FailedOperation.UnOpenError	服务未开通。
FailedOperation.Unknown	未知错误。
InvalidParameter.InvalidParameter	参数取值错误。
InvalidParameterValue.InvalidParameterValueLimit	参数值错误。
LimitExceeded.TooLargeFileError	文件太大。

错误码	描述
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。
ResourcesSoldOut.ChargeStatusException	计费状态异常。

图片智能裁剪

最近更新时间：2025-06-19 16:42:53

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

根据输入的裁剪比例，智能判断一张图片的最佳裁剪区域，确保原图的主体区域不受影响，以适应不同平台、设备的展示要求，避免简单拉伸带来的变形。

- 可前往 图像处理 产品文档中查看更多产品信息。
- 公共参数中的签名方式必须指定为V3版本，即配置SignatureMethod参数为TC3-HMAC-SHA256。

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：CropImage。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
Width	是	Integer	需要裁剪区域的宽度，与Height共同组成所需裁剪的图片宽高比例。 输入数字请大于0、小于图片宽度的像素值。 示例值：1100
Height	是	Integer	需要裁剪区域的高度，与Width共同组成所需裁剪的图片宽高比例。 输入数字请大于0、小于图片高度的像素值。 宽高比例（Width : Height）会简化为最简分数，即如果Width输入10、Height输入20，会简化为1：2。 Width : Height建议取值在[1, 2.5]之间，超过这个范围可能会影响效果。 示例值：880
ImageUrl	否	String	图片URL地址。 图片限制： • 图片格式：PNG、JPG、JPEG。 • 图片大小：所下载图片经Base64编码后不超过4M。图片下载时间不超过3秒。 建议： • 图片像素：大于50*50像素，否则影响识别效果。 • 长宽比：长边：短边<5。 接口响应时间会受到图片下载时间的影响，建议使用更可靠的存储服务，推荐将图片存储在腾讯云COS。 示例值：https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg
ImageBase64	否	String	图片经过Base64编码的内容。最大不超过4M。与ImageUrl同时存在时优先使用ImageUrl字段。 注意：图片需要Base64编码，并且要去掉编码头部。 示例值：/9j/4AAQSkZJRgABAQAAAQABAAD/4glo...lftXF/DjFZNXoSP5V2U0HMt/1FQf/Z

3. 输出参数

参数名称	类型	描述
X	Integer	裁剪区域左上角X坐标值 示例值：0
Y	Integer	裁剪区域左上角Y坐标值 示例值：110
Width	Integer	裁剪区域的宽度，单位为像素 示例值：1100
Height	Integer	裁剪区域的高度，单位为像素 示例值：880
OriginalWidth	Integer	原图宽度，单位为像素 示例值：1100
OriginalHeight	Integer	原图高度，单位为像素 示例值：1390
CropResult	Integer	0：抠图正常； 1：原图过长，指原图的高度是宽度的1.8倍以上； 2：原图过宽，指原图的宽度是高度的1.8倍以上； 3：抠图区域过长，指抠图的高度是主体备选框高度的1.6倍以上； 4：抠图区域过宽，指当没有检测到人脸时，抠图区域宽度是检测出的原图主体区域宽度的1.6倍以上； 5：纯色图，指裁剪区域视觉较为单一、缺乏主体部分； 6：宽高比异常，指Width : Height取值超出[1, 2.5]的范围； 以上是辅助决策的参考建议，可以根据业务需求选择采纳或忽视。 示例值：0
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 图像智能裁剪成功

输入一张图片，输出裁剪坐标结果

输入示例

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

{
"ImageUrl": "https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg",
"Width": 1100,
"Height": 880
}
```

输出示例

```
{
"Response": {
"X": 0,
"Y": 110,
"Width": 1100,
"Height": 880,
"OriginalWidth": 1100,
"OriginalHeight": 1390,
"CropResult": 0,
"RequestId": "bfd478e1-5c94-4e37-ad22-4a5224a09492"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.BalanceInsufficient	余额不足，开通失败，请充值后再开通。
FailedOperation.DownloadError	文件下载错误。
FailedOperation.ImageDecodeFailed	图片解码失败。
FailedOperation.ImageNotSupported	不支持的图片文件。
FailedOperation.InvokeChargeError	调用计费返回失败。
FailedOperation.RequestTimeout	后端服务超时。

错误码	描述
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
FailedOperation.UnKnowError	内部错误。
FailedOperation.UnOpenError	服务未开通。
FailedOperation.Unknown	未知错误。
InvalidParameter.InvalidParameter	参数取值错误。
InvalidParameterValue.InvalidParameterValueLimit	参数值错误。
LimitExceeded.TooLargeFileError	文件太大。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。
ResourcesSoldOut.ChargeStatusException	计费状态异常。

图像清晰度增强

最近更新时间：2025-06-19 16:42:53

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

传入一张图片，输出清晰度提升后的图片。

可以消除图片有损压缩导致的噪声，和使用滤镜、拍摄失焦导致的模糊。让图片的边缘和细节更加清晰自然。

- 可前往 图像处理 产品文档中查看更多产品信息。
- 公共参数中的签名方式必须指定为V3版本，即配置SignatureMethod参数为TC3-HMAC-SHA256。

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：EnhanceImage。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
ImageUrl	否	String	图片URL地址。 图片限制： • 图片格式：PNG、JPG、JPEG。 • 图片大小：所下载图片经Base64编码后不超过4M。图片下载时间不超过3秒。 建议： • 图片像素：大于50*50像素，最大不超过250万像素，否则影响识别效果。 • 长宽比：长边：短边<5。 接口响应时间会受到图片下载时间的影响，建议使用更可靠的存储服务，推荐将图片存储在腾讯云COS。 示例值：https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg
ImageBase64	否	String	支持PNG、JPG、JPEG、BMP，不支持 GIF 图片。图片经过Base64编码的内容。最大不超过4M。与ImageUrl同时存在时优先使用ImageUrl字段。 注意：图片需要Base64编码，并且要去掉编码头部。 示例值：/9j/4AAQSkZJRgABAQAAAQABAAD/4gIo...IftXF/DjFZNXoSP5V2U0HMt/1FQf/Z

3. 输出参数

参数名称	类型	描述
EnhancedImage	String	增强后图片的base64编码。 示例值：/9j/4AAQSkZJRgABAQAAAQABAAD/4gIo...IftXF/DjFZNXoSP5V2U0HMt/1FQf/Z

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

4. 示例

示例1 文件过大

输入示例

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

{
"ImageUrl": "https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg"
}
```

输出示例

```
{
"Response": {
"Error": {
"Code": "LimitExceeded.TooLargeFileError",
"Message": "文件太大"
},
"RequestId": "a1c397c8-5caf-44ce-975e-bf5e7c8242ef"
}
}
```

示例2 图像清晰度增强下载失败

输入示例

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

{
"ImageUrl": "https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg"
}
```

输出示例

```
{
"Response": {
"Error": {
"Code": "FailedOperation.DownLoadError",
"Message": "下载失败"
},
"RequestId": "97c5502b-ecfd-498a-84f2-28de25a3c71c"
}
}
```

示例3 图像清晰度增强请求成功

输入示例

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

{
"ImageUrl": "https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg"
}
```

输出示例

```
{
"Response": {
"EnhancedImage": "/9j/4AAQSkZJRgABAQAAAQABA...lftXF/DjFZNXoSP5V2U0HMt/1FQf/Z",
"RequestId": "75ad21c9-1db4-4032-9066-ab03e297349b"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

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

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.DownloadError	文件下载错误。
FailedOperation.ImageDecodeFailed	图片解码失败。
FailedOperation.ImageNotSupported	不支持的图片文件。
FailedOperation.InvokeChargeError	调用计费返回失败。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
FailedOperation.ServerError	算法服务异常，请重试。
FailedOperation.UnKnowError	内部错误。
FailedOperation.UnOpenError	服务未开通。
FailedOperation.Unknown	未知错误。
InvalidParameterValue.InvalidParameterValueLimit	参数值错误。
LimitExceeded.TooLargeFileError	文件太大。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.IsOpening	服务正在开通中，请稍等。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。
ResourcesSoldOut.ChargeStatusException	计费状态异常。

图像审核相关接口
恶心检测

最近更新时间：2025-06-19 16:42:53

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

输入一张图片，返回AI针对一张图片是否是恶心的一系列判断值。

通过恶心图片识别，可以判断一张图片是否令人恶心，同时给出它属于的潜在类别，让您能够过滤掉使人不愉快的图片。

- 公共参数中的签名方式必须指定为V3版本，即配置SignatureMethod参数为TC3-HMAC-SHA256。

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：DetectDisgust。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
ImageUrl	否	String	图片URL地址。 图片限制： • 图片格式：PNG、JPG、JPEG。 • 图片大小：所下载图片经Base64编码后不超过4M。图片下载时间不超过3秒。 建议： • 图片像素：大于50*50像素，否则影响识别效果； • 长宽比：长边：短边<5； 接口响应时间会受到图片下载时间的影响，建议使用更可靠的存储服务，推荐将图片存储在腾讯云COS。 示例值：https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg
ImageBase64	否	String	图片经过base64编码的内容。最大不超过4M。与ImageUrl同时存在时优先使用ImageUrl字段。 **注意：图片需要base64编码，并且要去掉编码头部。** 示例值：/9j/4AAQSkZJRgABAQAAAQABAAD/4gIo...IftXF/DjFZNXoSP5V2U0HMt/1FQf/Z

3. 输出参数

参数名称	类型	描述
Confidence	Float	对于图片中包含恶心内容的置信度，取值[0,1]，一般超过0.5则表明可能是恶心图片。 示例值：0.98

参数名称	类型	描述
Type	String	与图像内容最相似的恶心内容的类别，包含腐烂、密集、畸形、血腥、蛇、虫子、牙齿等。 示例值：蛇
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 恶心检测接口调用成功

输入示例

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

{
"ImageUrl": "https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg"
}
```

输出示例

```
{
"Response": {
"Confidence": 0.98,
"Type": "蛇",
"RequestId": "40afdbfc-be4c-4530-bb1a-3f1683b05dba"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation.DownloadError	文件下载错误。
FailedOperation.ImageDecodeFailed	图片解码失败。
FailedOperation.ImageNotSupported	不支持的图片文件。
FailedOperation.InvokeChargeError	调用计费返回失败。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
FailedOperation.UnKnowError	内部错误。
FailedOperation.UnOpenError	服务未开通。
InvalidParameter.InvalidParameter	参数取值错误。
InvalidParameterValue.InvalidParameterValueLimit	参数值错误。
LimitExceeded.TooLargeFileError	文件太大。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。
ResourcesSoldOut.ChargeStatusException	计费状态异常。

不良行为识别

最近更新时间：2025-06-19 16:42:53

1. 接口描述

接口请求域名： tiia.tencentcloudapi.com 。

可以识别输入的图片中是否包含不良行为，例如打架斗殴、赌博、抽烟等，可以应用于广告图、直播截图、短视频截图等审核，减少不良行为对平台内容质量的影响，维护健康向上的互联网环境。

- 公共参数中的签名方式必须指定为V3版本，即配置SignatureMethod参数为TC3-HMAC-SHA256。

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：DetectMisbehavior。
Version	是	String	公共参数，本接口取值：2019-05-29。
Region	是	String	公共参数，详见产品支持的 地域列表。
ImageUrl	否	String	图片URL地址。 图片限制： • 图片格式：PNG、JPG、JPEG。 • 图片大小：所下载图片经Base64编码后不超过4M。图片下载时间不超过3秒。 建议： • 图片像素：大于50*50像素，否则影响识别效果； • 长宽比：长边：短边<5； 接口响应时间会受到图片下载时间的影响，建议使用更可靠的存储服务，推荐将图片存储在腾讯云COS。 示例值：https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg
ImageBase64	否	String	图片经过base64编码的内容。最大不超过4M。与ImageUrl同时存在时优先使用ImageUrl字段。 **注意：图片需要base64编码，并且要去掉编码头部。** 示例值：/9j/4AAQSkZJRgABAQAAAQABAAD/4gIo...IftXF/DjFZNXoSP5V2U0HMt/1FQf/Z

3. 输出参数

参数名称	类型	描述
Confidence	Float	对于图片中包含不良行为的置信度，取值[0,1]，一般超过0.5则表明可能包含不良行为内容； 示例值：0.5
Type	String	图像中最可能包含的不良行为类别，包括赌博、打架斗殴、吸毒等。 示例值：赌博

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

4. 示例

示例1 不良行为识别调用成功

输入图片URL，识别不良行为信息。

输入示例

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

{
"ImageUrl": "https://liudhu-9527.cos.ap-guangzhou.myqcloud.com/input.jpeg"
}
```

输出示例

```
{
"Response": {
"Confidence": 0.5,
"Type": "赌博",
"RequestId": "7a975d0d-1ebe-4ee3-bde7-f1541fe5b135"
}
}
```

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.InvalidAuthorization	认证失败。
FailedOperation.DownloadError	文件下载错误。
FailedOperation.ImageDecodeFailed	图片解码失败。
FailedOperation.ImageNotSupported	不支持的图片文件。
FailedOperation.ImageSizeExceed	base64编码后的图片数据过大。
FailedOperation.InvokeChargeError	调用计费返回失败。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
FailedOperation.UnKnowError	内部错误。
FailedOperation.Unknown	未知错误。
InvalidParameter.InvalidParameter	参数取值错误。
InvalidParameterValue.InvalidParameterValueLimit	参数值错误。
LimitExceeded.TooLargeFileError	文件太大。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。

数据结构

最近更新时间：2025-06-19 16:42:54

Attribute

属性

被如下接口引用：CreateImage, SearchImage。

名称	类型	描述
Type	String	属性 示例值：灰色
Details	String	属性详情 示例值：颜色

AttributesForBody

属性检测到的人体

被如下接口引用：DetectChefDress, DetectSecurity。

名称	类型	描述
Rect	ImageRect	人体框。当不开启人体检测时，内部参数默认为0。 注意：此字段可能返回 null，表示取不到有效值。
DetectConfidence	Float	人体检测置信度。取值0-1之间，当不开启人体检测开关时默认为0。 示例值：0
Attributes	Array of BodyAttributes	属性信息。 注意：此字段可能返回 null，表示取不到有效值。

BodyAttributes

属性列表。

被如下接口引用：DetectChefDress, DetectSecurity。

名称	类型	描述
Label	String	属性值。 示例值：无厨师服
Confidence	Float	置信度，取值0-1之间。 示例值：0.608731
Name	String	属性名称。 示例值：厨师服识别

Box

图像主体区域。

被如下接口引用：CreateImage, SearchImage。

名称	类型	描述
Rect	ImageRect	图像主体区域。 示例值：{x:1,y:2,weight:1,height:2}
Score	Float	置信度。 示例值：1
CategoryId	Integer	主体区域类目ID 示例值：2

CarPlateContent

车牌信息

被如下接口引用：RecognizeCar, RecognizeCarPro。

名称	类型	描述
Plate	String	车牌信息。 示例值：浙G29999
Color	String	车牌颜色。 示例值：蓝色
Type	String	车牌类型，包含：0普通蓝牌，1双层黄牌，2单层黄牌，3新能源车牌，4使馆车牌，5领馆车牌，6澳门车牌，7香港车牌，8警用车牌，9教练车牌，10武警车牌，11军用车牌 -2遮挡污损模糊车牌/异常其他无牌 注意： 此字段可能返回 null，表示取不到有效值。 此字段结果遮挡污损模糊车牌/异常：包含PlateStatus参数的“遮挡污损模糊车牌”，针对车牌异常，建议参考此字段，更全面 示例值：普通蓝牌
PlateLocation	Array of Coord	车牌在图片中的坐标信息。 注意：此字段可能返回 null，表示取不到有效值。 示例值：[{"X":452,"Y":227},{"X":452,"Y":190},{"X":507,"Y":190},{"X":507,"Y":227}]
PlateStatus	String	判断车牌是否遮挡：“遮挡污损模糊车牌”和"正常车牌"。 示例值：正常车牌
PlateStatusConfidence	Integer	车牌遮挡的置信度，0-100。 示例值：99
PlateAngle	Float	车牌角度。 示例值：13.560508

CarTagItem

车辆属性识别的结果

被如下接口引用：RecognizeCar, RecognizeCarPro。

名称	类型	描述
Serial	String	车系 示例值：奔驰GLC级
Brand	String	车辆品牌 示例值：奔驰
Type	String	车辆类型

名称	类型	描述
		示例值：SUV
Color	String	车辆颜色 示例值：绿
Confidence	Integer	车系置信度，0-100 示例值：68
Year	Integer	年份，没识别出年份的时候返回0 示例值：0
CarLocation	Array of Coord	车辆在图片中的坐标信息 示例值：[{"X":87,"Y":291},{"X":87,"Y":1},{"X":540,"Y":1},{"X":540,"Y":291}]
PlateContent	CarPlateContent	车牌信息，仅车辆识别（增强版）支持 注意：此字段可能返回 null，表示取不到有效值。 示例值：{"Color":"蓝色","Plate":"浙G29XY9","PlateAngle":13.560508,"PlateLocation":[{"X":452,"Y":227},{"X":452,"Y":190},{"X":507,"Y":190},{"X":507,"Y":227}],"PlateStatus":"正常车牌","PlateStatusConfidence":99,"Type":"普通蓝牌"}
PlateConfidence	Integer	车牌信息置信度，0-100，仅车辆识别（增强版）支持 示例值：95
TypeConfidence	Integer	车辆类型置信度，0-100，仅车辆识别（增强版）支持 示例值：68
ColorConfidence	Integer	车辆颜色置信度，0-100，仅车辆识别（增强版）支持 示例值：99
Orientation	String	车辆朝向，仅车辆识别（增强版）支持 示例值：车头
OrientationConfidence	Integer	车辆朝向置信度，0-100，仅车辆识别（增强版）支持 示例值：90

ColorInfo

整张图颜色信息。

被如下接口引用：CreateImage, SearchImage。

名称	类型	描述
Color	String	RGB颜色值（16进制），例如：291A18。 示例值：4A3D46
Percentage	Float	当前颜色标签所占比例。 示例值：23.1
Label	String	颜色标签。蜜柚色，米驼色等。 示例值：蜜柚色

Coord

汽车坐标信息

被如下接口引用：RecognizeCar, RecognizeCarPro。

名称	类型	描述
X	Integer	横坐标x 示例值：231
Y	Integer	纵坐标y 示例值：200

DetectLabelItem

图像标签检测结果。

被如下接口引用：DetectLabel, DetectLabelPro。

名称	类型	描述
Name	String	图片中的物体名称。 示例值：string
Confidence	Integer	算法对于Name的置信度，0-100之间，值越高，表示对于Name越确定。 示例值：1
FirstCategory	String	标签的一级分类 示例值：string
SecondCategory	String	标签的二级分类 示例值：string

GroupInfo

图库信息。

被如下接口引用：DescribeGroups。

名称	类型	描述
GroupId	String	图库Id。 示例值：test_group_1
GroupName	String	图库名称。 示例值：图库001
Brief	String	图库简介。 示例值：简介
MaxCapacity	Integer	图库容量。 示例值：100
MaxQps	Integer	该库的访问限频 。 示例值：10
GroupType	Integer	图库类型，对应不同服务类型，默认为1。建议手动调整为4～6，1～3为历史版本，不推荐。 参数值： 4：在自建图库中搜索相同原图，可支持裁剪、翻转、调色、加水印后的图片搜索，适用于图片版权保护、原图查询等场景。 5：在自建图库中搜索相同或相似的商品图片，适用于商品分类、检索、推荐等电商场景。 6：在自建图片库中搜索与输入图片高度相似的图片，适用于相似图案、logo、纹理等图像元素的搜索。 示例值：1
PicCount	Integer	图库图片数量。 示例值：10

名称	类型	描述
CreateTime	String	图库创建时间。 示例值：2020-11-01 12:00:00
UpdateTime	String	图库更新时间。 示例值：2020-11-02 12:00:00

ImageInfo

图片信息

被如下接口引用：DescribeImages, SearchImage。

名称	类型	描述
EntityId	String	图片名称。 示例值：work-1
CustomContent	String	用户自定义的内容。 示例值：备注
Tags	String	图片自定义标签，JSON格式。 示例值：{"my_tag":"test"}
PicName	String	图片名称。 示例值：my_pic
Score	Integer	相似度。 示例值：90

ImageRect

图像主体区域坐标

被如下接口引用：CreateImage, DetectChefDress, DetectSecurity, SearchImage。

名称	类型	必选	描述
X	Integer	是	左上角横坐标。 示例值：200
Y	Integer	是	左上角纵坐标。 示例值：200
Width	Integer	是	宽度。 示例值：400
Height	Integer	是	高度。 示例值：400

ImageTag

图片标签。

被如下接口引用：DetectEnvelope。

名称	类型	描述
Name	String	标签内容。 示例值：文件封

名称	类型	描述
Confidence	Float	置信度范围在0~100之间。值越高，表示目标为相应结果的可能性越高。 示例值：99

ObjectInfo

图像的主体信息。

被如下接口引用：CreateImage, SearchImage。

名称	类型	描述
Box	Box	图像主体区域。 示例值：{}
CategoryId	Integer	主体类别ID。 示例值：2
Colors	Array of ColorInfo	整张图颜色信息。 示例值：{}
Attributes	Array of Attribute	属性信息。 示例值：{}
AllBox	Array of Box	图像的所有主体区域，置信度，以及主体区域类别ID。 注意：此字段可能返回 null，表示取不到有效值。 示例值：{}

Pet

宠物具体信息

被如下接口引用：DetectPet。

名称	类型	描述
Name	String	识别出的宠物类型（猫或者狗，暂不支持识别猫狗品种）。 示例值：cat
Score	Integer	识别服务给识别目标打出的置信度，范围在0~100之间。值越高，表示目标为相应结果的可能性越高。 示例值：99
Location	Rect	识别目标在图片中的坐标。

Product

检测到的单个商品结构体

被如下接口引用：DetectProduct。

名称	类型	描述
Name	String	图片中商品的三级分类识别结果，选取所有三级分类中的置信度最大者 示例值：男士西服套装
Parents	String	三级商品分类对应的一级分类和二级分类，两级之间用“-”（中划线）隔开，例如商品名称是“硬盘”，那么Parents输出为“电脑、办公-电脑配件” 示例值：服饰内衣-男装
Confidence	Integer	算法对于Name的置信度，0~100之间，值越高，表示对于Name越确定

名称	类型	描述
		示例值：80
XMin	Integer	商品坐标X轴的最小值 示例值：336
YMin	Integer	商品坐标Y轴的最小值 示例值：191
XMax	Integer	商品坐标X轴的最大值 示例值：799
YMax	Integer	商品坐标Y轴的最大值 示例值：775

Rect

具体坐标，可用来判断边界

被如下接口引用：CreateImage, DetectPet。

名称	类型	必选	描述
X	Integer	是	x轴坐标 示例值：100
Y	Integer	是	y轴坐标 示例值：100
Width	Integer	是	(x,y)坐标距离长度 示例值：200
Height	Integer	是	(x,y)坐标距离高度 示例值：200

错误码

最近更新时间：2024-12-23 11:03:55

功能说明

如果返回结果中存在 Error 字段，则表示调用 API 接口失败。例如：

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

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

错误码列表

公共错误码

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

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

业务错误码

错误码	说明
FailedOperation.BalanceInsufficient	余额不足，开通失败，请充值后再开通。
FailedOperation.DownLoadError	文件下载失败。
FailedOperation.DownloadError	文件下载错误。
FailedOperation.EmptyImageError	图片内容为空。
FailedOperation.GroupCountExceeded	图库数量超出限制。
FailedOperation.ImageDecodeFailed	图片解码失败。
FailedOperation.ImageDeleteFailed	图片删除失败。

错误码	说明
FailedOperation.ImageDownloadError	图片下载错误。
FailedOperation.ImageEntityCountExceed	超出Entity数量限制。
FailedOperation.ImageGroupChargeStatusClose	停止服务，控制台开关关闭。
FailedOperation.ImageGroupEmpty	图库为空。
FailedOperation.ImageNotFoundInfo	未找到图片信息。
FailedOperation.ImageNotSupported	不支持的图片文件。
FailedOperation.ImageNumExceed	超出图库限制。
FailedOperation.ImageResolutionExceed	图片分辨率过大。
FailedOperation.ImageResolutionInsufficient	图片分辨率过小。
FailedOperation.ImageSearchInvalid	未查询到结果。
FailedOperation.ImageSizeExceed	base64编码后的图片数据过大。
FailedOperation.ImageUnQualified	图片不满足检测要求。
FailedOperation.ImageUrlInvalid	url地址不合法，无法下载。
FailedOperation.InnerError	内部错误。
FailedOperation.InvokeChargeError	调用计费返回失败。
FailedOperation.NoBodyInPhoto	图片中没有人体。
FailedOperation.NoObjectDetected	未检测到目标。
FailedOperation.ParameterEmpty	参数为空。
FailedOperation.RecognizeFailded	车辆识别失败。
FailedOperation.RequestError	后端服务请求失败。
FailedOperation.RequestTimeout	后端服务超时。
FailedOperation.RpcFail	RPC请求失败，一般为算法微服务故障。
FailedOperation.ServerError	算法服务异常，请重试。
FailedOperation.TooLargeFileError	文件太大。
FailedOperation.UnKnowError	内部错误。
FailedOperation.UnOpenError	服务未开通。
FailedOperation.Unknown	未知错误。
InvalidParameter.ImageFormatNotSupport	不支持的图片格式。
InvalidParameter.InvalidParameter	参数取值错误。
InvalidParameter.PictureSolidColorError	图片不可为纯色图。
InvalidParameterValue.BriefTooLong	图库简介过长。
InvalidParameterValue.CustomContentTooLong	自定义内容过长。
InvalidParameterValue.EntityIdEmpty	物品ID为空。

错误码	说明
InvalidParameterValue.EntityIdTooLong	物品ID超出长度限制。
InvalidParameterValue.FilterInvalid	Filter参数不合法。
InvalidParameterValue.FilterSizeExceed	Filter参数过长。
InvalidParameterValue.ImageEmpty	图片为空。
InvalidParameterValue.ImageGroupIdAlreadyExist	图库ID已存在。
InvalidParameterValue.ImageGroupIdIllegal	图库ID不合法。
InvalidParameterValue.ImageGroupIdNotExist	图库ID不存在。
InvalidParameterValue.ImageGroupIdTooLong	图库ID超出长度限制。
InvalidParameterValue.ImageGroupNameEmpty	图库名称为空。
InvalidParameterValue.ImageGroupNameTooLong	图库名称超出长度限制。
InvalidParameterValue.InvalidParameterValueLimit	参数值错误。
InvalidParameterValue.LimitExceed	返回数量不在合法范围内。
InvalidParameterValue.PicNameAlreadyExist	图片名称重复。
InvalidParameterValue.PicNameEmpty	图片名称为空。
InvalidParameterValue.PicNameInvalid	PicName不合法。
InvalidParameterValue.PicNameTooLong	图片名称超出长度限制。
InvalidParameterValue.TagsKeysExceed	标签数量过多。
InvalidParameterValue.TagsValueIllegal	标签值类型不合法。
InvalidParameterValue.TagsValueSizeExceed	标签值长度过长。
InvalidParameterValue.UrlIllegal	URL格式不合法。
LimitExceeded.TooLargeFileError	文件太大。
MissingParameter.ErrorParameterEmpty	必选参数为空。
ResourceUnavailable.InArrears	账号已欠费。
ResourceUnavailable.IsOpening	服务正在开通中，请稍等。
ResourceUnavailable.NotExist	计费状态未知，请确认是否已在控制台开通服务。
ResourceUnavailable.StopUsing	帐号已停服。
ResourcesSoldOut.ChargeStatusException	计费状态异常。