

内容安全

API 文档

产品文档



腾讯云

【版权声明】

©2013-2020 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100。

文档目录

API 文档

更新历史

简介

API 概览

调用方式

请求结构

公共参数

签名方法 v3

签名方法

返回结果

自定义识别相关接口

查询文本样本库

删除文本样本库

新增文本样本库

查询图片样本库

删除图片样本库

新增图片样本库

文本内容安全相关接口

文本内容检测

图片内容安全相关接口

图片内容检测

内容安全相关接口

人工审核对外接口

数据结构

错误码

API 文档

更新历史

最近更新时间：2020-09-15 08:03:14

第 15 次发布

发布时间：2020-07-30 08:07:34

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- [TextData](#)
 - 新增成员：DataId, Extra

第 14 次发布

发布时间：2020-07-24 08:07:51

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [ManualReview](#)

新增数据结构：

- [ManualReviewContent](#)
- [ManualReviewData](#)

修改数据结构：

- [RiskDetails](#)
 - 新增成员：Label

第 13 次发布

发布时间：2020-07-17 08:07:40

本次发布包含了以下内容：

改善已有的文档。

新增数据结构：

- [RiskDetails](#)

修改数据结构：

- [TextData](#)
 - 新增成员：RiskDetails

第 12 次发布

发布时间：2020-07-09 08:07:45

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- [TextModeration](#)
 - 新增入参：Device, User

新增数据结构：

- [Device](#)
- [User](#)

修改数据结构：

- [CustomResult](#)
 - 新增成员：Type

第 11 次发布

发布时间：2020-05-27 08:08:35

本次发布包含了以下内容：

改善已有的文档。

删除接口：

- CommonMediaRecognition

删除数据结构：

- CallBackAck
- CallBackData
- ContentMsgGjh
- MediaGjh
- MediaInfoTypeGjh
- MessageCommonGjh
- MessageContentGjh

- MessageIdGjh
- UserInfoGjh

第 10 次发布

发布时间：2020-05-22 20:13:51

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- CommonMediaRecognition

新增数据结构：

- CallBackAck
- CallBackData
- ContentMsgGjh
- MediaGjh
- MediaInfoTypeGjh
- MessageCommonGjh
- MessageContentGjh
- MessageIdGjh
- UserInfoGjh

第 9 次发布

发布时间：2020-05-15 08:07:21

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- [TextModeration](#)
 - 新增入参：BizType, DataId, SdkAppId

新增数据结构：

- [Coordinate](#)
- [CustomResult](#)
- [DetailResult](#)
- [OCRItem](#)

修改数据结构：

- [OCRDetect](#)

- 新增成员：Item
- [TextData](#)
 - 新增成员：CustomResult, DetailResult, BizType, EvilLabel, Score, Suggestion

第 8 次发布

发布时间：2020-05-11 10:03:36

本次发布包含了以下内容：

改善已有的文档。

新增数据结构：

- [PhoneDetect](#)

修改数据结构：

- [ImageData](#)
 - 新增成员：PhoneDetect

第 7 次发布

发布时间：2020-03-12 22:16:03

本次发布包含了以下内容：

改善已有的文档。

删除接口：

- AudioModeration
- DescribeModerationOverview
- VideoModeration

删除数据结构：

- OverviewRecord

第 6 次发布

发布时间：2020-03-10 00:51:35

本次发布包含了以下内容：

改善已有的文档。

新增数据结构：

- [TextOutputComm](#)
- [TextOutputID](#)

- [TextOutputRes](#)

修改数据结构：

- [TextData](#)
 - 新增成员：Common, ID, Res

第 5 次发布

发布时间：2020-02-27 20:16:44

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- [CreateTextSample](#)
 - 新增入参：Test
 - 新增出参：ErrMsg

新增数据结构：

- [Logo](#)
- [LogoDetail](#)
- [RectF](#)

修改数据结构：

- [FileSample](#)
 - 新增成员：CompressFileUrl
- [FileSampleInfo](#)
 - 新增成员：CompressFileUrl
- [ImageData](#)
 - 新增成员：LogoDetect
- [ImagePolityDetect](#)
 - 新增成员：PolityLogoDetail

第 4 次发布

发布时间：2019-10-11 14:21:52

本次发布包含了以下内容：

改善已有的文档。

新增数据结构：

- [CodeDetail](#)
- [CodeDetect](#)

- [CodePosition](#)

修改数据结构：

- [ImageData](#)
 - 新增成员：CodeDetect

第 3 次发布

发布时间：2019-08-15 23:21:19

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [CreateFileSample](#)
- [DeleteFileSample](#)
- [DescribeFileSample](#)

新增数据结构：

- [FileSample](#)
- [FileSampleInfo](#)
- [OCRDetect](#)

修改数据结构：

- [ImageData](#)
 - 新增成员：OCRDetect

第 2 次发布

发布时间：2019-06-21 18:58:55

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [CreateTextSample](#)
- [DeleteTextSample](#)
- [DescribeTextSample](#)

新增数据结构：

- [Filter](#)
- [ImageHotDetect](#)
- [TextSample](#)

修改数据结构：

- [ImageData](#)
 - 新增成员：HotDetect
- [ImagePolityDetect](#)
 - 新增成员：PolityItems

第 1 次发布

发布时间：2019-04-26 16:49:17

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- AudioModeration
- DescribeModerationOverview
- [ImageModeration](#)
- [TextModeration](#)
- VideoModeration

新增数据结构：

- [ImageData](#)
- [ImageIllegalDetect](#)
- [ImagePolityDetect](#)
- [ImagePornDetect](#)
- [ImageTerrorDetect](#)
- OverviewRecord
- [Similar](#)
- [TextData](#)

简介

最近更新时间：2019-06-21 19:12:46

内容安全 API 升级到 3.0 版本。全新的 API 接口文档更加规范和全面，统一的参数风格和公共错误码，统一的 SDK/CLI 版本与 API 文档严格一致，给您带来简单快捷的使用体验。支持全地域就近接入让您更快连接腾讯云产品。

内容安全基于腾讯海量数据及深度识别技术支撑，可高效准确地识别文字、图片和视频等多媒体的违规内容，帮助客户降低色情、暴恐、涉政、广告及其他客户自定义的违规风险，解决网站和 App 开发者的违规识别难题。节约人工审核成本的同时，大幅提高识别效率，保护业务健康发展。

API 概览

最近更新时间：2020-09-08 09:50:19

图片内容安全相关接口

接口名称	接口功能
ImageModeration	图片内容检测

内容安全相关接口

接口名称	接口功能
ManualReview	人工审核对外接口

文本内容安全相关接口

接口名称	接口功能
TextModeration	文本内容检测

自定义识别相关接口

接口名称	接口功能
CreateFileSample	新增图片样本库
CreateTextSample	新增文本样本库
DeleteFileSample	删除图片样本库
DeleteTextSample	删除文本样本库
DescribeFileSample	查询图片样本库
DescribeTextSample	查询文本样本库

调用方式

请求结构

最近更新时间：2020-09-28 08:03:53

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

接入地域	域名
欧洲地区(莫斯科)	cms.eu-moscow.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：2020-10-23 08:03:59

公共参数是用于标识用户和接口签名的参数，如非必要，在每个接口单独的接口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。

签名方法 v3

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 [文档](#)。

注意：接口文档中的示例由于目的是展示接口参数用法，简化起见，使用的是签名方法 v1 GET 请求，如果依旧想使用签名方法 v1 请参考下文章节。

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：

参数名称	类型	必选	描述
X-TC-Action	String	是	操作的接口名称。取值参考接口文档中输入参数公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。
X-TC-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。
X-TC-Timestamp	Integer	是	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。
X-TC-Version	String	是	操作的 API 的版本。取值参考接口文档中输入公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
Authorization	String	是	HTTP 标准身份认证头部字段，例如： TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024 其中， - TC3-HMAC-SHA256：签名方法，目前固定取该值； - Credential：签名凭证，AKIDEXAMPLE 是 SecretId；Date 是 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致；service 为产品名，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 cms； - 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 。
X-TC-Token	String	否	临时证书所用的 Token，需要结合临时密钥一起使用。临时密钥和 Token 需要到访问管理服务调用接口获取。长期密钥不需要 Token。

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

HTTP GET 请求结构示例：

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

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

HTTP POST (application/json) 请求结构示例：

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

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

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

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

10
--58731222010402--
```

签名方法 v1

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输入公共参数 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	临时证书所用的 Token，需要结合临时密钥一起使用。临时密钥和 Token 需要到访问管理服务调用接口获取。长期密钥不需要 Token。

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

HTTP GET 请求结构示例：

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

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

HTTP POST 请求结构示例：

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

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

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

地域列表

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

地域	取值
华北地区(北京)	ap-beijing
华南地区(广州)	ap-guangzhou
港澳台地区(中国香港)	ap-hongkong
亚太南部(孟买)	ap-mumbai
华东地区(上海)	ap-shanghai
亚太东南(新加坡)	ap-singapore
亚太东北(东京)	ap-tokyo
欧洲地区(法兰克福)	eu-frankfurt
美国东部(弗吉尼亚)	na-ashburn
美国西部(硅谷)	na-siliconvalley
北美地区(多伦多)	na-toronto

签名方法 v3

最近更新时间：2020-09-08 09:50:19

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，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 7 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++](#)。

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请求参数中指定该签名结果并以指定的方式和格式发送请求。

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】即可以创建一对密钥。

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。

假设用户的 SecretId 和 SecretKey 分别是：`AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****` 和 `Gu5t9xGARNpq86cd98joQYCN3*****`。用户想查看广州云服务器名为“未命名”的主机状态，只返回一条数据。则请求可能为：

```
curl -X POST https://cvm.tencentcloudapi.com \
-H "Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host, Signature=2230eefd229f582d8b1b891af7107b91597240707d778ab3738f756258d7652c" \
-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，字符集 UTF8，推荐使用编程语言标准库，所有特殊字符均需编码，大写形式。
CanonicalHeaders	参与签名的头部信息，至少包含 host 和 content-type 两个头部，也可加入自定义的头部参与签名以提高自身请求的唯一性和安全性。 拼接规则： 1. 头部 key 和 value 统一转成小写，并去掉首尾空格，按照 key:value\n 格式拼接； 2. 多个头部，按照头部 key（小写）的 ASCII 升序进行拼接。 此示例计算结果是 content-type:application/json; charset=utf-8\nhost:cvm.tencentcloudapi.com\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。

字段名称	解释
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
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

content-type;host
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。

字段名称	解释
HashedCanonicalRequest	前述步骤拼接所得规范请求串的哈希值，计算伪代码为 <code>Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))</code> 。此示例计算结果是 <code>5ffe6a04c0664d6b969fab9a13bdab201d63ee709638e2749d62a09ca18d7031</code> 。

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，可能导致运行一段时间后，请求必定失败，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
5ffe6a04c0664d6b969fab9a13bdab201d63ee709638e2749d62a09ca18d7031
```

3. 计算签名

1) 计算派生签名密钥，伪代码如下：

```
SecretKey = "Gu5t9xGARNpq86cd98joQYCN3*****"
SecretDate = HMAC_SHA256("TC3" + SecretKey, Date)
SecretService = HMAC_SHA256(SecretDate, Service)
SecretSigning = HMAC_SHA256(SecretService, "tc3_request")
```

派生出的密钥 SecretDate、SecretService 和 SecretSigning 是二进制的数，可能包含不可打印字符，此处不展示中间结果。

请注意，不同的编程语言，HMAC 库函数中参数顺序可能不一样，此处的伪代码密钥参数在后，请以实际编程语言为准。通常标准库函数会提供二进制格式的计算值，也即此处使用的，也会提供打印友好的十六进制格式的计算值，将在下面计算签名结果时使用。

字段名称	解释
SecretKey	原始的 SecretKey，即 <code>Gu5t9xGARNpq86cd98joQYCN3*****</code> 。
Date	即 Credential 中的 Date 字段信息。此示例取值为 <code>2019-02-25</code> 。
Service	即 Credential 中的 Service 字段信息。此示例取值为 <code>cvm</code> 。

2) 计算签名，伪代码如下：

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

此示例计算结果是 `2230eefd229f582d8b1b891af7107b91597240707d778ab3738f756258d7652c`。

4. 拼接 Authorization

按如下格式拼接 Authorization：

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

字段名称	解释
Algorithm	签名方法，固定为 TC3-HMAC-SHA256。
SecretId	密钥对中的 SecretId，即 AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****。
CredentialScope	见上文，凭证范围。此示例计算结果是 2019-02-25/cvm/tc3_request。
SignedHeaders	见上文，参与签名的头部信息。此示例取值为 content-type;host。
Signature	签名值。此示例计算结果是 2230eefd229f582d8b1b891af7107b91597240707d778ab3738f756258d7652c。

根据以上规则，示例中得到的值为：

```
TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****/2019-02-25/cvm/tc3_request, SignedHeaders  
=content-type;host, Signature=2230eefd229f582d8b1b891af7107b91597240707d778ab3738f756258d7652c
```

最终完整的调用信息如下：

```
POST https://cvm.tencentcloudapi.com/  
Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****/2019-02-25/cvm/tc3_request,  
SignedHeaders=content-type;host, Signature=2230eefd229f582d8b1b891af7107b91597240707d778ab3738f756258d765  
2c  
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"}] }
```

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++](#)

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

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.DataMapper;

public class TencentCloudAPITC3Demo {
    private final static Charset UTF8 = StandardCharsets.UTF_8;
    private final static String SECRET_ID = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****";
    private final static String SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3*****";
    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 DataMapper.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";
String signedHeaders = "content-type;host";

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  
secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****"  
secret_key = "Gu5t9xGARNpq86cd98joQYCN3*****"  
  
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.datetime.fromtimestamp(timestamp).strftime("%Y-%m-%d")  
params = {"Limit": 1, "Filters": [{"Name": "instance-name", "Values": [u"未命名"]}]}  
  
# ***** 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\n" % (ct, host)  
signed_headers = "content-type;host"  
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"
    "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() {
    secretId := "AKIDz8krbsJ5yKBZQpn74WFkmlPx3*****"
    secretKey := "Gu5t9xGARNpq86cd98joQYCN3*****"
    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 := "content-type:application/json; charset=utf-8\n" + "host:" + host + "\n"
signedHeaders := "content-type;host"
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"\n\n-H "X-TC-Timestamp: %d"\n\n-H "X-TC-Version: %s"\n\n-H "X-TC-Region: %s"\n\n-d '%s', host, authorization, host, action, timestamp, version, region, payload)\nfmt.Println(curl)\n}
```

PHP

```
<?php\n$secretId = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****";\n$secretKey = "Gu5t9xGARNpq86cd98joQYCN3*****";\n$host = "cvm.tencentcloudapi.com";\n$service = "cvm";\n$version = "2017-03-12";\n$action = "DescribeInstances";\n$region = "ap-guangzhou";\n// $timestamp = time();\n$timestamp = 1551113065;\n$algorithm = "TC3-HMAC-SHA256";\n\n// step 1: build canonical request string\n$httpRequestMethod = "POST";\n$canonicalUri = "/";\n$canonicalQueryString = "";\n$canonicalHeaders = "content-type:application/json; charset=utf-8\n"."host:". $host. "\n";\n$signedHeaders = "content-type;host";\n$payload = '{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]'};\n$hashedRequestPayload = hash("SHA256", $payload);\n$canonicalRequest = $httpRequestMethod. "\n"\n. $canonicalUri. "\n"\n. $canonicalQueryString. "\n"\n. $canonicalHeaders. "\n"\n. $signedHeaders. "\n"\n. $hashedRequestPayload;\necho $canonicalRequest.PHP_EOL;\n\n// step 2: build string to sign\n$date = gmdate("Y-m-d", $timestamp);\n$credentialScope = $date. "/" . $service. "/tc3_request";\n$hashedCanonicalRequest = hash("SHA256", $canonicalRequest);\n$stringToSign = $algorithm. "\n"\n. $timestamp. "\n"\n. $credentialScope. "\n"\n. $hashedCanonicalRequest;\necho $stringToSign.PHP_EOL;\n\n// step 3: sign string\n$secretDate = hash_hmac("SHA256", $date, "TC3". $secretKey, true);\n$secretService = hash_hmac("SHA256", $service, $secretDate, true);\n$secretSigning = hash_hmac("SHA256", "tc3_request", $secretService, true);\n$signature = hash_hmac("SHA256", $stringToSign, $secretSigning);
```

```
echo $signature.PHP_EOL;

// step 4: build authorization
$authorization = $algorithm
." Credential=".$secretId."/".$credentialScope
.", SignedHeaders=content-type;host, Signature=".$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
secret_id = 'AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****'
secret_key = 'Gu5t9xGARNpq86cd98joQYCN3*****'

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}\n"
signed_headers = 'content-type;host'
# 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 *****, we hard-code it here.
payload = '{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}'
```

```
hashed_request_payload = Digest::SHA256.hexdigest(payload)
canonical_request = [
http_request_method,
canonical_uri,
canonical_querystring,
canonical_headers,
signed_headers,
hashed_request_payload,
].join("\n")

puts canonical_request

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

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

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

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

DotNet

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

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

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

    public static Dictionary<String, String> BuildHeaders(string secretid,
        string secretkey, string service, string endpoint, string region,
        string action, string version, DateTime date, string requestPayload)
    {
        string datestr = date.ToString("yyyy-MM-dd");
        DateTime startTime = new DateTime(1970, 1, 1, 0, 0, 0, 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";
        string signedHeaders = "content-type;host";
        string hashedRequestPayload = SHA256Hex(requestPayload);
        string canonicalRequest = httpRequestMethod + "\n"
            + canonicalUri + "\n"
            + canonicalQueryString + "\n"
            + canonicalHeaders + "\n"
            + signedHeaders + "\n"
            + hashedRequestPayload;
        Console.WriteLine(canonicalRequest);
        Console.WriteLine("-----");

        // ***** 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);
Console.WriteLine("-----");

// ***** 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);
Console.WriteLine("-----");

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

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
    string SECRET_ID = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****";
    string SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3*****";

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

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

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

    Console.WriteLine("POST https://cvm.tencentcloudapi.com");
```

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

NodeJS

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

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

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

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

function main(){
    // 密钥参数
    const SECRET_ID = "AKIDz8krbsJ5yKBZQpn74WFkmlPx3*****"
    const SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3*****"

    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 signedHeaders = "content-type;host"

    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"
```

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

// ***** 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)
console.log("-----")

// ***** 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)
console.log("-----")

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

const Call_Information = '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(Call_Information)
}
main()
```

C++

```
#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++)
    {
        sprintf(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);

#ifdef 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
    string SECRET_ID = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****";
    string SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3*****";

    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 canonicalHeaders = "content-type:application/json; charset=utf-8\nhost:" + host + "\n";
string signedHeaders = "content-type;host";
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;
cout << "-----" << endl;

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

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

string headers = "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 << headers << endl;
return 0;
};
```

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符合，也有可能是密钥 SecretKey 错误导致的。
AuthFailure.TokenFailure	临时证书 Token 错误。
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）。

签名方法

最近更新时间：2020-08-11 08:10:17

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

首次接触，建议使用 [API Explorer](#) 中的“签名串生成”功能，选择签名版本为“API 3.0 签名 v1”，可以生成签名过程进行验证，并提供了部分编程语言的签名示例，也可直接生成 SDK 代码。推荐使用腾讯云 API 配套的 7 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](#)。

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

1. 申请安全凭证

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

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

申请安全凭证的具体步骤如下：

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

注意：每个账号最多可以拥有两对 SecretId/SecretKey。

2. 生成签名串

有了安全凭证 SecretId 和 SecretKey 后，就可以生成签名串了。以下是生成签名串的详细过程：

假设用户的 SecretId 和 SecretKey 分别是：

- SecretId: AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****
- SecretKey: Gu5t9xGARNpq86cd98joQYCN3*****

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

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

参数名称	中文	参数值
Action	方法名	DescribeInstances
SecretId	密钥 ID	AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****
Timestamp	当前时间戳	1465185768

参数名称	中文	参数值
Nonce	随机正整数	11886
Region	实例所在区域	ap-guangzhou
InstanceId.0	待查询的实例 ID	ins-09dx96dg
Offset	偏移量	0
Limit	最大允许输出	20
Version	接口版本号	2017-03-12

2.1. 对参数排序

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

```
{
  'Action': 'DescribeInstances',
  'InstanceId.0': 'ins-09dx96dg',
  'Limit': 20,
  'Nonce': 11886,
  'Offset': 0,
  'Region': 'ap-guangzhou',
  'SecretId': 'AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****',
  'Timestamp': 1465185768,
  'Version': '2017-03-12',
}
```

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

2.2. 拼接请求字符串

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

然后将格式化后的各个参数用 "&" 拼接在一起，最终生成的请求字符串为：

```
Action=DescribeInstances&InstanceId.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****&Timestamp=1465185768&Version=2017-03-12
```

2.3. 拼接签名原字符串

此步骤生成签名原字符串。签名原字符串由以下几个参数构成：

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

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

示例的拼接结果为：

```
GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceId=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmlPx3*****&Timestamp=1465185768&Version=2017-03-12
```

2.4. 生成签名串

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

具体代码如下，以 PHP 语言为例：

```
$secretKey = 'Gu5t9xGARNpq86cd98joQYCN3*****';
$srcStr = 'GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceId=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmlPx3*****&Timestamp=1465185768&Version=2017-03-12';
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));
echo $signStr;
```

最终得到的签名串为：

```
zmmjn35mikh6pM3V7sUEuX4wyYM=
```

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

3. 签名串编码

生成的签名串并不能直接作为请求参数，需要对其进行 URL 编码。

如上一步生成的签名串为 zmmjn35mikh6pM3V7sUEuX4wyYM=，最终得到的签名串请求参数（Signature）为：zmmjn35mikh6pM3V7sUEuX4wyYM%3D，它将用于生成最终的请求 URL。

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

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

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

4. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理。

错误代码	错误描述
------	------

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

5. 签名演示

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

- [Python](#)
- [Java](#)
- [PHP](#)
- [Go](#)
- [NodeJS](#)
- [.NET](#)

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

最终输出的 url 可能为：`https://cvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceId=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****&Signature=zmmjn35mikh6pM3V7sUEuX4wyYM%3D&Timestamp=1465185768&Version=2017-03-12`。

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

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

注意：以下代码仅适用于 API 3.0，不能直接用于其他的签名流程，即使是旧版的 API，由于存在细节差异也会导致签名计算错误，请以对应的实际文档为准。

Java

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

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

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

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

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

    public static void main(String[] args) throws Exception {
        TreeMap<String, Object> params = new TreeMap<String, Object>(); // TreeMap可以自动排序
        // 实际调用时应当使用随机数，例如：params.put("Nonce", new Random().nextInt(java.lang.Integer.MAX_VALUE));
        params.put("Nonce", 11886); // 公共参数
        // 实际调用时应当使用系统当前时间，例如：params.put("Timestamp", System.currentTimeMillis() / 1000);
        params.put("Timestamp", 1465185768); // 公共参数
        params.put("SecretId", "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****"); // 公共参数
        params.put("Action", "DescribeInstances"); // 公共参数
        params.put("Version", "2017-03-12"); // 公共参数
        params.put("Region", "ap-guangzhou"); // 公共参数
        params.put("Limit", 20); // 业务参数
        params.put("Offset", 0); // 业务参数
        params.put("InstanceId.0", "ins-09dx96dg"); // 业务参数
        params.put("Signature", sign(getStringToSign(params), "Gu5t9xGARNpq86cd98joQYCN3*****", "HmacSHA1")); // 公共参
        数
        System.out.println(getUrl(params));
    }
}
```

Python

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

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

import requests

secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3*****"

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

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

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

Golang

```
package main

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

```
func main() {
secretId := "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****"
secretKey := "Gu5t9xGARNpq86cd98joQYCN3*****"
params := map[string]string{
    "Nonce": "11886",
    "Timestamp": "1465185768",
    "Region": "ap-guangzhou",
    "SecretId": secretId,
    "Version": "2017-03-12",
    "Action": "DescribeInstances",
    "InstanceId.0": "ins-09dx96dg",
    "Limit": "20",
    "Offset": "0",
}

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

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

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

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

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

PHP

```
<?php
$secretId = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****";
$secretKey = "Gu5t9xGARNpq86cd98joQYCN3*****";
$params["Nonce"] = 11886;//rand();
$params["Timestamp"] = 1465185768;//time();
$params["Region"] = "ap-guangzhou";
$params["SecretId"] = $secretId;
```

```
$param["Version"] = "2017-03-12";
$param["Action"] = "DescribeInstances";
$param["InstanceId.0"] = "ins-09dx96dg";
$param["Limit"] = 20;
$param["Offset"] = 0;

ksort($param);

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

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

Ruby

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

secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3*****"

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

```
params << "#{item[0]}=#{item[1]}"
end
sign += params.join('&')
digest = OpenSSL::Digest.new('sha1')
data['Signature'] = Base64.encode64(OpenSSL::HMAC.digest(digest, secret_key, sign))
puts data['Signature']

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

DotNet

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

public class Application {
    public static string Sign(string signKey, string secret)
    {
        string signRet = string.Empty;
        using (HMACSHA1 mac = new HMACSHA1(Encoding.UTF8.GetBytes(signKey)))
        {
            byte[] hash = mac.ComputeHash(Encoding.UTF8.GetBytes(secret));
            signRet = Convert.ToBase64String(hash);
        }
        return signRet;
    }

    public static string MakeSignPlainText(SortedDictionary<string, string> requestParams, string requestMethod, string requestHost, string requestPath)
    {
        string retStr = "";
        retStr += requestMethod;
        retStr += requestHost;
        retStr += requestPath;
        retStr += "?";
        string v = "";
        foreach (string key in requestParams.Keys)
        {
            v += string.Format("{0}={1}&", key, requestParams[key]);
        }
        retStr += v.TrimEnd('&');
        return retStr;
    }

    public static void Main(string[] args)
    {
        // 密钥参数
```



```
string SECRET_ID = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****";
string SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3*****";

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

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

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

NodeJS

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

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

function formatSignString(reqMethod, endpoint, path, strParam){
    let strSign = reqMethod + endpoint + path + "?" + strParam.slice(1);
    return strSign;
}

function sha1(secretKey, strsign){
    let signMethodMap = {'HmacSHA1': 'sha1'};
    let hmac = crypto.createHmac(signMethodMap['HmacSHA1'], secretKey || "");
    return hmac.update(Buffer.from(strsign, 'utf8')).digest('base64')
}

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

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

function main(){
// 密钥参数
const SECRET_ID = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*****"
const SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3*****"

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

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

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

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

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

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

返回结果

最近更新时间：2020-06-12 08:09:12

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，可以联系我们，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，可以联系我们，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，下表列出了公共错误码。

错误码	错误描述
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。
AuthFailure.SignatureExpire	签名过期。
AuthFailure.SignatureFailure	签名错误。
AuthFailure.TokenFailure	token 错误。
AuthFailure.UnauthorizedOperation	请求未 CAM 授权。
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。
InvalidParameterValue	参数取值错误。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数错误。
NoSuchVersion	接口版本不存在。
RequestLimitExceeded	请求的次数超过了频率限制。
ResourceInUse	资源被占用。
ResourceInsufficient	资源不足。
ResourceNotFound	资源不存在。
ResourceUnavailable	资源不可用。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。
UnsupportedProtocol	HTTPS 请求方法错误，只支持 GET 和 POST 请求。

错误码	错误描述
UnsupportedRegion	接口不支持所传地域。

自定义识别相关接口

查询文本样本库

最近更新时间：2020-07-13 08:19:38

1. 接口描述

接口请求域名：cms.tencentcloudapi.com。

本文档适用于文本内容安全、音频内容安全自定义识别库的管理。

支持批量查询文本样本库。

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：DescribeTextSample。
Version	是	String	公共参数，本接口取值：2019-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表 ，本接口仅支持其中的: ap-guangzhou
Filters.N	否	Array of Filter	支持通过标签值进行筛选
Limit	否	Integer	数量限制，默认为20，最大值为100
Offset	否	Integer	偏移量，默认为0
OrderDirection	否	String	升序（asc）还是降序（desc），默认：desc
OrderField	否	String	按某个字段排序，目前仅支持CreatedAt排序

3. 输出参数

参数名称	类型	描述
TextSampleSet	Array of TextSample	符合要求的样本的信息
TotalCount	Integer	符合要求的样本的数量

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询文本样本库

查询白库

输入示例

```
https://cms.tencentcloudapi.com/?Action=DescribeTextSampleInstances
&Offset=0
&Limit=20
&OrderField=CreatedAt
&OrderDirection=desc
&Filters.0.Name=Label
&Filters.0.Value=1
&<公共请求参数>
```

输出示例

```
{
  "Response": {
    "RequestId": "c1935701-668a-4e0f-9f25-7e88a5f3f7af",
    "TotalCount": 10,
    "TextSampleSet": [
      {
        "Label": 1,
        "EvilType": 20002,
        "Content": "太子党",
        "Code": 0,
        "Status": 1,
        "CreatedAt": "2019-05-28 17:40:01"
      },
      {
        "Label": 1,
        "EvilType": 20002,
        "Content": "赵家党",
        "Code": 0,
        "Status": 1,
        "CreatedAt": "2019-05-28 17:40:01"
      }
    ]
  },
  "retcode": 0,
  "retmsg": "success"
}
```

5. 开发者资源

API Explorer

该工具提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力，能显著降低使用云 API 的难度，推荐使用。

- [API 3.0 Explorer](#)

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](#)
- [Tencent Cloud SDK 3.0 for Java](#)
- [Tencent Cloud SDK 3.0 for PHP](#)
- [Tencent Cloud SDK 3.0 for Go](#)
- [Tencent Cloud SDK 3.0 for NodeJS](#)
- [Tencent Cloud SDK 3.0 for .NET](#)

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
InternalServerError	内部错误

删除文本样本库

最近更新时间：2020-07-13 08:19:39

1. 接口描述

接口请求域名：cms.tencentcloudapi.com。

本文档适用于文本内容安全、音频内容安全自定义识别库的管理。

删除文本样本库，暂时只支持单个删除。

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：DeleteTextSample。
Version	是	String	公共参数，本接口取值：2019-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表 ，本接口仅支持其中的: ap-guangzhou
Ids.N	是	Array of String	唯一标识数组，目前暂时只支持单个删除

3. 输出参数

参数名称	类型	描述
Progress	Integer	任务状态 1：已完成 2：处理中
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 删除文本样本库

删除文本样本库

输入示例

```
https://cms.tencentcloudapi.com/?Action=DeleteTextSample
Ids.0=QUOdt1dK16
&<公共请求参数>
```

输出示例

```
{
  "Response": {
    "RequestId": "c1935701-668a-4e0f-9f25-7e88a5f3f7af",
    "Progress": 1
  },
  "retcode": 0,
  "retmsg": "success"
}
```

5. 开发者资源

API Explorer

该工具提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力，能显著降低使用云 API 的难度，推荐使用。

- [API 3.0 Explorer](#)

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](#)
- [Tencent Cloud SDK 3.0 for Java](#)
- [Tencent Cloud SDK 3.0 for PHP](#)
- [Tencent Cloud SDK 3.0 for Go](#)
- [Tencent Cloud SDK 3.0 for NodeJS](#)
- [Tencent Cloud SDK 3.0 for .NET](#)

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
InternalServerError	内部错误

新增文本样本库

最近更新时间：2020-07-13 08:19:40

1. 接口描述

接口请求域名：cms.tencentcloudapi.com。

本文档适用于文本内容安全、音频内容安全自定义识别库的管理。

通过该接口可以将文本新增到样本库。

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：CreateTextSample。
Version	是	String	公共参数，本接口取值：2019-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表 ，本接口仅支持其中的: ap-guangzhou
Contents.N	是	Array of String	关键词数组
EvilType	是	Integer	恶意类型 100：正常 20001：政治 20002：色情 20006：涉毒违法 20007：谩骂 24001：暴恐 20105：广告引流
Label	是	Integer	样本类型 1：黑库 2：白库
Test	否	String	测试修改参数

3. 输出参数

参数名称	类型	描述
ErrMsg	String	操作样本失败时返回的错误信息示例："样本1":错误码,"样本2":错误码
Progress	Integer	任务状态 1：已完成 2：处理中
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 新增文本样本

将“测试”关键词加入自定义词库，并设置该词类型为正常

输入示例

```
https://cms.tencentcloudapi.com/?Action=CreateTextSample
&Label=2
&EvilType=100&Contents.0=测试
&<公共请求参数>
```

输出示例

```
{
  "Response": {
    "RequestId": "c1935701-668a-4e0f-9f25-7e88a5f3f7af",
    "Progress": 2,
    "ErrMsg": "3:-1009,4:-1009,"
  },
  "retcode": 0,
  "retmsg": "success"
}
```

5. 开发者资源

API Explorer

该工具提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力，能显著降低使用云 API 的难度，推荐使用。

- [API 3.0 Explorer](#)

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](#)
- [Tencent Cloud SDK 3.0 for Java](#)
- [Tencent Cloud SDK 3.0 for PHP](#)

- [Tencent Cloud SDK 3.0 for Go](#)
- [Tencent Cloud SDK 3.0 for NodeJS](#)
- [Tencent Cloud SDK 3.0 for .NET](#)

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
InternalServerError	内部错误

查询图片样本库

最近更新时间：2020-07-13 08:19:39

1. 接口描述

接口请求域名：cms.tencentcloudapi.com。

本文档适用于图片内容安全、视频内容安全自定义识别库的管理。

查询图片样本库，支持批量查询。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

[点击调试](#)

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeFileSample。
Version	是	String	公共参数，本接口取值：2019-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表 ，本接口仅支持其中的: ap-guangzhou
Filters.N	否	Array of Filter	支持通过标签值进行筛选
Limit	否	Integer	数量限制，默认为20，最大值为100
Offset	否	Integer	偏移量，默认为0
OrderDirection	否	String	升序（asc）还是降序（desc），默认：desc
OrderField	否	String	按某个字段排序，目前仅支持CreatedAt排序

3. 输出参数

参数名称	类型	描述
FileSampleSet	Array of FileSampleInfo	符合要求的样本的信息
TotalCount	Integer	符合要求的样本的数量
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询图片样本库

查询黑图片样本库前20条内容，并按CreatedAt倒序排序

输入示例

```
https://cms.tencentcloudapi.com/?Action=DescribeFileSample
&Offset=0
&Limit=20
&OrderField=CreatedAt
&OrderDirection=desc
&Filters.0.Name=Label
&Filters.0.Value=1
&<公共请求参数>
```

输出示例

```
{
  "Response": {
    "RequestId": "c1935701-668a-4e0f-9f25-7e88a5f3f7af",
    "TotalCount": 10,
    "FileSampleSet": [
      {
        "Label": 1,
        "EvilType": 20002,
        "FileName": "1.png",
        "FileType": "image",
        "FileMd5": "",
        "Code": 0,
        "Status": 1,
        "CreatedAt": "2019-05-28 17:40:01"
      },
      {
        "Label": 1,
        "EvilType": 20002,
        "FileName": "2.png",
        "FileType": "image",
        "FileMd5": "",
        "Code": 0,
        "Status": 1,
        "CreatedAt": "2019-05-28 17:40:01"
      }
    ]
  }
}
```

5. 开发者资源

API Explorer

该工具提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力，能显著降低使用云 API 的难度，推荐使用。

- [API 3.0 Explorer](#)

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](#)
- [Tencent Cloud SDK 3.0 for Java](#)
- [Tencent Cloud SDK 3.0 for PHP](#)
- [Tencent Cloud SDK 3.0 for Go](#)
- [Tencent Cloud SDK 3.0 for NodeJS](#)
- [Tencent Cloud SDK 3.0 for .NET](#)

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
InternalServerError	内部错误

删除图片样本库

最近更新时间：2020-07-13 08:19:39

1. 接口描述

接口请求域名：cms.tencentcloudapi.com。

本文档适用于图片内容安全、视频内容安全自定义识别库的管理。

删除图片样本库，支持批量删除，一次提交不超过20个。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

[点击调试](#)

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DeleteFileSample。
Version	是	String	公共参数，本接口取值：2019-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表 ，本接口仅支持其中的: ap-guangzhou
Ids.N	是	Array of String	唯一标识数组

3. 输出参数

参数名称	类型	描述
Progress	Integer	任务状态 1：已完成 2：处理中
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 删除图片样本库

删除图片样本库

输入示例

```
https://cms.tencentcloudapi.com/?Action=DeleteFileSample
&Ids.0=QUOdt1dK16
&<公共请求参数>
```

输出示例

```
{
  "Response": {
    "RequestId": "c1935701-668a-4e0f-9f25-7e88a5f3f7af",
    "Progress": 1
  }
}
```

5. 开发者资源

API Explorer

该工具提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力，能显著降低使用云 API 的难度，推荐使用。

- [API 3.0 Explorer](#)

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](#)
- [Tencent Cloud SDK 3.0 for Java](#)
- [Tencent Cloud SDK 3.0 for PHP](#)
- [Tencent Cloud SDK 3.0 for Go](#)
- [Tencent Cloud SDK 3.0 for NodeJS](#)
- [Tencent Cloud SDK 3.0 for .NET](#)

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
InternalServerError	内部错误

新增图片样本库

最近更新时间：2020-07-13 08:19:40

1. 接口描述

接口请求域名：cms.tencentcloudapi.com。

本文档适用于图片内容安全、视频内容安全自定义识别库的管理。

通过该接口可以将图片新增到样本库。

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：CreateFileSample。
Version	是	String	公共参数，本接口取值：2019-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表 ，本接口仅支持其中的: ap-guangzhou
Contents.N	是	Array of FileSample	文件类型结构数组
EvilType	是	Integer	恶意类型 100：正常 20001：政治 20002：色情 20006：涉毒违法 20007：谩骂 24001：暴恐 20105：广告引流
FileType	是	String	image：图片
Label	是	Integer	样本类型 1：黑库 2：白库

3. 输出参数

参数名称	类型	描述
Progress	Integer	任务状态 1：已完成 2：处理中
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 新增图片类型样本

将一张图片新增到样本库中，标注为色情

输入示例

```
https://cms.tencentcloudapi.com/?Action=CreateFileSample
&EvilType=20002
&Label=1
&FileType.0=image
&Contents.0.FileName=苍老师.jpg
&Contents.0.FileUrl=http://example.com/q.jpg
&Contents.0.FileMd5=9cc56fe973ec99c1f03afcc920df3a0b
&<公共请求参数>
```

输出示例

```
{
  "Response": {
    "RequestId": "c1935701-668a-4e0f-9f25-7e88a5f3f7af",
    "Progress": 2
  }
}
```

5. 开发者资源

API Explorer

该工具提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力，能显著降低使用云 API 的难度，推荐使用。

- [API 3.0 Explorer](#)

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](#)
- [Tencent Cloud SDK 3.0 for Java](#)
- [Tencent Cloud SDK 3.0 for PHP](#)
- [Tencent Cloud SDK 3.0 for Go](#)

- [Tencent Cloud SDK 3.0 for NodeJS](#)
- [Tencent Cloud SDK 3.0 for .NET](#)

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
InternalServerError	内部错误

文本内容安全相关接口

文本内容检测

最近更新时间：2020-07-13 08:19:40

1. 接口描述

接口请求域名：cms.tencentcloudapi.com。

文本内容检测（Text Moderation）服务使用了深度学习技术，识别涉黄、涉政、涉恐等有害内容，同时支持用户配置词库，打击自定义的违规文本。

默认接口请求频率限制：500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：TextModeration。
Version	是	String	公共参数，本接口取值：2019-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表 。
Content	是	String	文本内容Base64编码。原文长度需小于15000字节，即5000个汉字以内。
Device	否	Device	设备相关信息
User	否	User	用户相关信息
BizType	否	Integer	该字段用于标识业务场景。您可以在内容安全控制台创建对应的ID，配置不同的内容审核策略，通过接口调用，默认不填为0，后端使用默认策略
DataId	否	String	数据ID，英文字母、下划线、-组成，不超过64个字符
SdkAppId	否	Integer	业务应用ID

3. 输出参数

参数名称	类型	描述
------	----	----

参数名称	类型	描述
Data	TextData	识别结果
BusinessCode	Integer	业务返回码
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 文本内容检测

输入示例

```
https://cms.tencentcloudapi.com/?Action=TextModeration
&Content=57uY5aOw57uY6lmy
&<公共请求参数>
```

输出示例

```
{
  "Response": {
    "BusinessCode": 0,
    "Data": {
      "EvilType": 20001,
      "EvilFlag": 1,
      "BizType": 2,
      "Score": 72,
      "EvilLabel": "Polity",
      "Suggestion": "Review",
      "Keywords": [
        "傻子"
      ],
      "DetailResult": [
        {
          "EvilType": 20001,
          "KeywordsList": [
            "傻子"
          ],
          "Prediction": 72
        }
      ],
      "RequestId": "jlUM5m23JWDeAX1tQirYT9cP"
    },
    "retmsg": "success",
    "retcode": 0
  }
}
```

5. 开发者资源

API Explorer

该工具提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力，能显著降低使用云 API 的难度，推荐使用。

- [API 3.0 Explorer](#)

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](#)
- [Tencent Cloud SDK 3.0 for Java](#)
- [Tencent Cloud SDK 3.0 for PHP](#)
- [Tencent Cloud SDK 3.0 for Go](#)
- [Tencent Cloud SDK 3.0 for NodeJS](#)
- [Tencent Cloud SDK 3.0 for .NET](#)

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
InternalServerError	内部错误
InvalidParameter.ParameterError	参数不可用
InvalidParameterValue.ErrTextContentType	文本内容类型错误：需base64编码
ResourceUnavailable.ErrTextTimeOut	文本识别服务超时
UnauthorizedOperation.ErrAuth	鉴权失败
UnauthorizedOperation.Unauthorized	未开通权限

图片内容安全相关接口

图片内容检测

最近更新时间：2020-07-13 08:19:41

1. 接口描述

接口请求域名：cms.tencentcloudapi.com。

图片内容检测服务（Image Moderation, IM）能自动扫描图片，识别涉黄、涉恐、涉政、涉毒等有害内容，同时支持用户配置图片黑名单，打击自定义的违规图片。

默认接口请求频率限制：50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：ImageModeration。
Version	是	String	公共参数，本接口取值：2019-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表 。
FileContent	否	String	文件内容 Base64,与FileUrl必须二选一
FileMD5	否	String	文件MD5值
FileUrl	否	String	文件地址

3. 输出参数

参数名称	类型	描述
Data	ImageData	识别结果
BusinessCode	Integer	业务返回码
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 图片内容检测

输入示例

```
https://cms.tencentcloudapi.com/?Action=ImageModeration
&FileMD5=字符串型
&FileContent=字符串型
&FileUrl=https://xxx.jpg
&<公共请求参数>
```

输出示例

```
{
  "Response": {
    "BusinessCode": 0,
    "Data": {
      "EvilFlag": 0,
      "EvilType": 100,
      "CodeDetect": {
        "ModerationCode": 0,
        "ModerationDetail": [
          {
            "CodeCharset": "UTF-8",
            "CodePosition": [
              {
                "FloatX": 5.8333335,
                "FloatY": 314.16666
              },
              {
                "FloatX": 5.8333335,
                "FloatY": 5.8333335
              },
              {
                "FloatX": 314.16666,
                "FloatY": 5.8333335
              },
              {
                "FloatX": 314.16666,
                "FloatY": 314.16666
              }
            ],
            "CodeText": "http://xxx",
            "CodeType": 2
          }
        ],
        "HotDetect": {
          "EvilType": 100,
          "HitFlag": 0,
          "Keywords": [],
```

```
"Labels": [],
"Score": 0
},
"IlllegalDetect": {
  "EvilType": 100,
  "HitFlag": 0,
  "Keywords": [],
  "Labels": [],
  "Score": 0
},
"OCRDetect": {
  "TextInfo": ""
},
"PolityDetect": {
  "EvilType": 100,
  "FaceNames": [],
  "HitFlag": 0,
  "Keywords": [],
  "PolityItems": [],
  "Score": 0
},
"PornDetect": {
  "EvilType": 100,
  "HitFlag": 0,
  "Keywords": [],
  "Labels": [],
  "Score": 0
},
"Similar": {
  "EvilType": 100,
  "HitFlag": 0,
  "SeedUrl": ""
},
"TerrorDetect": {
  "EvilType": 100,
  "HitFlag": 0,
  "Keywords": [],
  "Labels": [],
  "Score": 0
}
},
"RequestId": "57937de8-84a6-452d-b723-0cb6dad0dd1e"
},
"retcode": 0,
"retmsg": "ok"
}
```

5. 开发者资源

API Explorer

该工具提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力，能显著降低使用云 API 的难度，推荐使用。

- [API 3.0 Explorer](#)

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](#)
- [Tencent Cloud SDK 3.0 for Java](#)
- [Tencent Cloud SDK 3.0 for PHP](#)
- [Tencent Cloud SDK 3.0 for Go](#)
- [Tencent Cloud SDK 3.0 for NodeJS](#)
- [Tencent Cloud SDK 3.0 for .NET](#)

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
InternalServerError	内部错误
InvalidParameter.ParameterError	参数不可用
InvalidParameterValue.ErrFileContent	FileContent不可用：需Base64编码
InvalidParameterValue.InvalidImageContent	图片内容错误
MissingParameter.ErrFileUrl	FileUrl或FileContent都为空
ResourceNotFound.ErrDowdownInternalError	文件链接下载内部错误
ResourceNotFound.ErrDowdownParamsError	文件链接下载服务参数异常
ResourceNotFound.ErrDowdownSourceError	文件链接下载源错误
ResourceNotFound.ErrDowdownTimeOut	文件链接下载超时
ResourceUnavailable.ErrImageTimeOut	图片识别服务超时
UnauthorizedOperation.ErrAuth	鉴权失败
UnauthorizedOperation.Unauthorized	未开通权限

内容安全相关接口

人工审核对外接口

最近更新时间：2020-07-24 08:07:53

1. 接口描述

接口请求域名：cms.tencentcloudapi.com。

人工审核对外接口

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

[点击调试](#)

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：ManualReview。
Version	是	String	公共参数，本接口取值：2019-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表 ，本接口仅支持其中的: ap-guangzhou
ReviewContent	是	ManualReviewContent	人工审核信息

3. 输出参数

参数名称	类型	描述
Data	ManualReviewData	人审接口同步响应结果
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 提交审核任务

业务侧将疑似违规的音频、视频、文本、图片等数据，通过此接口提交到腾讯审核平台，由后端专业人工审核团队进行审核校验。

输入示例

```
https://cms.tencentcloudapi.com/?Action=ManualReview
&ReviewContent.ContentId="123456"
&ReviewContent.Content="文本为Base64，其他类型提交URL"
&ReviewContent.ContentType=3
&ReviewContent.AutoResult=0
&ReviewContent.AutoDetailCode=20001
&ReviewContent.BatchId="20200419_0001"
&ReviewContent.UserInfo="{}"
&ReviewContent.CreateTime=130000
&ReviewContent.Priority=2
&ReviewContent.CallBackInfo=""
&ReviewContent.Title=""
&<公共请求参数>
```

输出示例

```
{
  "Response": {
    "RequestId": "153e746d-d323-4042-b8ab-50b56ad54d36",
    "Data": {
      "ContentId": "123456",
      "BatchId": "20200419_0001"
    }
  }
}
```

5. 开发者资源

API Explorer

该工具提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力，能显著降低使用云 API 的难度，推荐使用。

- [API 3.0 Explorer](#)

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](#)
- [Tencent Cloud SDK 3.0 for Java](#)
- [Tencent Cloud SDK 3.0 for PHP](#)
- [Tencent Cloud SDK 3.0 for Go](#)
- [Tencent Cloud SDK 3.0 for NodeJS](#)
- [Tencent Cloud SDK 3.0 for .NET](#)

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 [公共错误码](#)。

错误码	描述
InvalidParameterValue.CustomAppId	渠道AppId参数错误
InvalidParameterValue.DuplicateContentID	重复提交ContentID
InvalidParameterValue.ErrAction	Action参数值错误
InvalidParameterValue.ErrAppId	AppId参数值错误
InvalidParameterValue.ErrRequestID	RequestID参数值错误
InvalidParameterValue.ErrRequestSource	RequestSource请求参数错误或请求失败
InvalidParameterValue.ErrUin	Uin参数值错误
InvalidParameterValue.InvalidBatchId	BatchId取值错误
InvalidParameterValue.InvalidContent	Content参数错误
InvalidParameterValue.InvalidContentID	ContentID参数错误
InvalidParameterValue.InvalidContentType	ContentType取值错误 1 图片 2 视频 3 文本 4 音频
InvalidParameterValue.InvalidCustomAppId	渠道AppId参数错误
InvalidParameterValue.InvalidParameter	参数取值错误
InvalidParameterValue.InvalidPriority	Priority取值错误
InvalidParameterValue.InvalidTitle	Title取值错误
UnauthorizedOperation.ErrAuth	鉴权失败

数据结构

最近更新时间：2020-10-29 08:03:10

CodeDetail

从图片中检测到的二维码，可能为多个

被如下接口引用：ImageModeration。

名称	类型	必选	描述
CodePosition	Array of CodePosition	否	二维码在图片中的位置，由边界点的坐标表示
CodeCharset	String	否	二维码文本的编码格式
CodeText	String	否	二维码的文本内容
CodeType	Integer	否	二维码的类型：1:ONED_BARCODE，2:QRCOD，3:WXCODE，4:PDF417，5:DATAMATRIX

CodeDetect

图片二维码详情

被如下接口引用：ImageModeration。

名称	类型	必选	描述
ModerationDetail	Array of CodeDetail	否	从图片中检测到的二维码，可能为多个
ModerationCode	Integer	否	检测是否成功，0：成功，-1：出错

CodePosition

二维码在图片中的位置，由边界点的坐标表示

被如下接口引用：ImageModeration。

名称	类型	必选	描述
FloatX	Float	否	二维码边界点X轴坐标
FloatY	Float	否	二维码边界点Y轴坐标

Coordinate

坐标

被如下接口引用：ImageModeration。

名称	类型	必选	描述
Cx	Integer	否	左上角横坐标
Cy	Integer	否	左上角纵坐标
Height	Integer	否	高度
Width	Integer	否	宽度

CustomResult

文本返回的自定义词库结果

被如下接口引用：TextModeration。

名称	类型	必选	描述
Keywords	Array of String	否	命中的自定义关键词
LibId	String	否	自定义库id
LibName	String	否	自定义词库名称
Type	String	否	命中的自定义关键词的类型

DetailResult

文本返回的详细结果

被如下接口引用：TextModeration。

名称	类型	必选	描述
EvilLabel	String	否	恶意标签，Normal：正常，Polity：涉政，Porn：色情，Illegal：违法，Abuse：谩骂，Terror：暴恐，Ad：广告，Custom：自定义关键词
EvilType	Integer	否	恶意类型 100：正常 20001：政治 20002：色情 20006：涉毒违法 20007：谩骂 20105：广告引流 24001：暴恐
Keywords	Array of String	否	该标签下命中的关键词

名称	类型	必选	描述
Score	Integer	否	该标签模型命中的分值

Device

设备信息

被如下接口引用：TextModeration。

名称	类型	必选	描述
DeviceId	String	否	设备指纹ID
IDFA	String	否	IOS设备，Identifier For Advertising（广告标识符）
IDFV	String	否	IOS设备，IDFV - Identifier For Vendor（应用开发商标识符）
IMEI	String	否	设备序列号
IP	String	否	用户IP
Mac	String	否	Mac地址
TokenId	String	否	设备指纹Token

FileSample

文件类型样本

被如下接口引用：CreateFileSample。

名称	类型	必选	描述
FileMd5	String	是	文件md5
FileName	String	是	文件名称
FileUrl	String	是	文件url
CompressFileUrl	String	否	文件压缩后云url

FileSampleInfo

文件样本返回信息

被如下接口引用：DescribeFileSample。

名称	类型	必选	描述
----	----	----	----

名称	类型	必选	描述
Code	Integer	是	处理错误码
CreatedAt	Integer	是	创建时间戳
EvilType	Integer	是	恶意类型 100：正常 20001：政治 20002：色情 20006：涉毒违法 20007：谩骂 24001：暴恐
FileMd5	String	是	文件的md5
FileName	String	是	文件名称
FileType	String	是	文件类型
Id	String	是	唯一标识
Label	Integer	是	样本类型 1：黑库 2：白库
Status	Integer	是	任务状态 1：添加完成 2：添加处理中 3：下载中 4：下载完成 5：上传完成 6：步骤完成
CompressFileUrl	String	否	文件压缩后云url
FileUrl	String	否	文件的url

Filter

筛选数据结构

被如下接口引用：DescribeFileSample, DescribeTextSample。

名称	类型	必选	描述
Name	String	是	需要过滤的字段
Value	String	是	需要过滤字段的值

ImageData

图片识别结果详情

被如下接口引用：ImageModeration。

名称	类型	必选	描述
EvilFlag	Integer	是	是否恶意 0：正常 1：可疑
EvilType	Integer	是	恶意类型 100：正常 20001：政治 20002：色情 20006：涉毒违法 20007：谩骂 20103：性感 24001：暴恐
CodeDetect	CodeDetect	否	图片二维码详情
HotDetect	ImageHotDetect	否	图片性感详情
IllegalDetect	ImageIllegalDetect	否	图片违法详情
LogoDetect	LogoDetail	否	logo详情
OCRDetect	OCRDetect	否	图片OCR详情
PhoneDetect	PhoneDetect	否	手机检测详情
PolityDetect	ImagePolityDetect	否	图片涉政详情
PornDetect	ImagePornDetect	否	图片涉黄详情
Similar	Similar	否	图片相似度详情
TerrorDetect	ImageTerrorDetect	否	图片暴恐详情

ImageHotDetect

图片性感详情

被如下接口引用：ImageModeration。

名称	类型	必选	描述
EvilType	Integer	是	恶意类型 100：正常 20103：性感
HitFlag	Integer	是	处置判定 0：正常 1：可疑
Keywords	Array of String	否	关键词明细
Labels	Array of String	否	性感标签：性感特征中文描述

名称	类型	必选	描述
Score	Integer	否	性感分：分值范围 0-100，分数越高性感倾向越明显

ImageIllegalDetect

图片违法详情

被如下接口引用：ImageModeration。

名称	类型	必选	描述
EvilType	Integer	是	恶意类型 100：正常 20006：涉毒违法
HitFlag	Integer	是	处置判定 0：正常 1：可疑
Keywords	Array of String	否	关键词明细
Labels	Array of String	否	违法标签：返回违法特征中文描述，如赌桌，枪支
Score	Integer	否	违法分：分值范围 0-100，分数越高违法倾向越明显

ImagePolityDetect

图片涉政详情

被如下接口引用：ImageModeration。

名称	类型	必选	描述
EvilType	Integer	是	恶意类型 100：正常 20001：政治
HitFlag	Integer	是	处置判定 0：正常 1：可疑
PolityLogoDetail	Array of Logo	否	命中的logo标签信息
FaceNames	Array of String	否	命中的人脸名称
Keywords	Array of String	否	关键词明细
PolityItems	Array of String	否	命中的政治物品名称
Score	Integer	否	政治（人脸）分：分值范围 0-100，分数越高可疑程度越高

ImagePornDetect

图片涉黄详情

被如下接口引用：ImageModeration。

名称	类型	必选	描述
EvilType	Integer	是	恶意类型 100：正常 20002：色情
HitFlag	Integer	是	处置判定 0：正常 1：可疑
Keywords	Array of String	否	关键词明细
Labels	Array of String	否	色情标签：色情特征中文描述
Score	Integer	否	色情分：分值范围 0-100，分数越高色情倾向越明显

ImageTerrorDetect

图片暴恐详情

被如下接口引用：ImageModeration。

名称	类型	必选	描述
EvilType	Integer	否	恶意类型 100：正常 24001：暴恐
HitFlag	Integer	否	处置判定 0：正常 1：可疑
Keywords	Array of String	否	关键词明细
Labels	Array of String	否	暴恐标签：返回暴恐特征中文描述
Score	Integer	否	暴恐分：分值范围0--100，分数越高暴恐倾向越明显

Logo

Logo

被如下接口引用：ImageModeration。

名称	类型	必选	描述
RrectF	RrectF	否	logo图标坐标信息
Confidence	Float	否	logo图标置信度
Name	String	否	logo图标名称

LogoDetail

LogoDetail

被如下接口引用：ImageModeration。

名称	类型	必选	描述
AppLogoDetail	Array of Logo	否	命中的Applogo详情

ManualReviewContent

人审审核数据相关信息

被如下接口引用：ManualReview。

名称	类型	必选	描述
BatchId	String	是	审核批次号
Content	String	是	审核内容
ContentId	String	是	消息Id
ContentType	Integer	是	审核内容类型 1 图片 2 视频 3 文本 4 音频
UserInfo	User	否	用户信息
AutoDetailCode	Integer	否	机器审核类型，与腾讯机器审核定义一致 100 正常 20001 政治 20002 色情 20006 违法 20007 谩骂 24001 暴恐 20105 广告 20103 性感
AutoResult	Integer	否	机器审核结果 0 放过 1 拦截
CallBackInfo	String	否	回调信息标识，回传数据时原样返回
CreateTime	String	否	创建时间 格式“2020-01-01 00:00:12”
Priority	Integer	否	审核优先级，可选值 [1,2,3,4]，其中 1 最高，4 最低
Title	String	否	标题

ManualReviewData

人工审核接口返回结果，由ContentId和BatchId组成

被如下接口引用：ManualReview。

名称	类型	必选	描述
BatchId	String	是	人审内容批次号
ContentId	String	是	人审内容ID

OCRDetect

OCR识别结果详情

被如下接口引用：ImageModeration。

名称	类型	必选	描述
Item	Array of OCRItem	否	识别到的详细信息
TextInfo	String	否	识别到的文本信息

OCRItem

OCR详情

被如下接口引用：ImageModeration。

名称	类型	必选	描述
TextPosition	Coordinate	否	检测到的文本坐标信息
EvilLabel	String	否	文本命中具体标签
EvilType	Integer	否	文本命中恶意违规类型
Keywords	Array of String	否	文本命中违规的关键词
Rate	Integer	否	文本涉嫌违规分值
TextContent	String	否	检测到的文本信息

PhoneDetect

手机模型识别检测

被如下接口引用：ImageModeration。

名称	类型	必选	描述
----	----	----	----

名称	类型	必选	描述
EvilType	Integer	否	恶意类型 100：正常 21000：综合
HitFlag	Integer	否	处置判定 0：正常 1：可疑
Labels	Array of String	否	特征中文描述
Score	Integer	否	分值范围 0-100，分数越高倾向越明显

RiskDetails

账号风险检测结果

被如下接口引用：TextModeration。

名称	类型	必选	描述
Keywords	Array of String	否	预留字段，暂时不使用
Label	String	否	风险类别，RiskAccount，RiskIP, RiskIMEI
Lable	String	否	预留字段，暂时不用
Level	Integer	否	风险等级，1:疑似，2：恶意

RrectF

logo位置信息

被如下接口引用：ImageModeration。

名称	类型	必选	描述
Cx	Float	否	logo横坐标
Cy	Float	否	logo纵坐标
Height	Float	否	logo图标高度
Rotate	Float	否	logo图标中心旋转度
Width	Float	否	logo图标宽度

Similar

相似度详情

被如下接口引用：ImageModeration。

名称	类型	必选	描述
EvilType	Integer	是	恶意类型 100：正常 20001：政治 20002：色情 20006：涉毒违法 20007：谩骂 24001：暴恐
HitFlag	Integer	是	处置判定 0：未匹配到 1：恶意 2：白样本
SeedUrl	String	是	返回的种子url

TextData

文本识别结果详情

被如下接口引用：TextModeration。

名称	类型	必选	描述
EvilFlag	Integer	是	是否恶意 0：正常 1：可疑
EvilType	Integer	是	恶意类型 100：正常 20001：政治 20002：色情 20006：涉毒违法 20007：谩骂 20105：广告引流 24001：暴恐
Common	TextOutputComm	否	消息类公共相关参数
CustomResult	Array of CustomResult	否	返回的自定义词库结果
DetailResult	Array of DetailResult	否	返回的详细结果
ID	TextOutputID	否	消息类ID信息
Res	TextOutputRes	否	消息类输出结果
RiskDetails	Array of RiskDetails	否	账号风险检测结果
BizType	Integer	否	最终使用的BizType
DataId	String	否	和请求中的DataId一致，原样返回
EvilLabel	String	否	恶意标签，Normal：正常，Polity：涉政，Porn：色情，Illegal：违法， Abuse：谩骂，Terror：暴恐，Ad：广告，Custom：自定义关键词

名称	类型	必选	描述
Extra	String	否	输出的其他信息，不同客户内容不同
Keywords	Array of String	否	命中的关键词
Score	Integer	否	命中的模型分值
Suggestion	String	否	建议值,Block：打击,Review：待复审,Normal：正常

TextOutputComm

消息类输出公共参数

被如下接口引用：TextModeration。

名称	类型	必选	描述
AppID	Integer	否	接入业务的唯一ID
BUCtrlID	Integer	否	接口唯一ID，旁路调用接口返回有该字段，标识唯一接口
SendTime	Integer	否	消息发送时间
Uin	Integer	否	请求字段里的Common.Uin

TextOutputID

消息类输出ID参数

被如下接口引用：TextModeration。

名称	类型	必选	描述
MsgID	String	否	接入业务的唯一ID
Uin	String	否	用户账号uin，对应请求协议里的Content.User.Uin。旁路结果有回带，串联结果无该字段

TextOutputRes

消息类输出结果参数

被如下接口引用：TextModeration。

名称	类型	必选	描述
Operator	String	否	操作人,信安处理人企业微信ID
ResultCode	Integer	否	恶意操作码， 删除（1），通过（2），先审后发（100012）

名称	类型	必选	描述
ResultMsg	String	否	操作结果备注说明
ResultType	Integer	否	恶意类型，广告（10001），政治（20001），色情（20002），社会事件（20004），暴力（20011），低俗（20012），违法犯罪（20006），欺诈（20008），版权（20013），谣言（20104），其他（21000）

TextSample

文字样本信息

被如下接口引用：DescribeTextSample。

名称	类型	必选	描述
Code	Integer	是	处理错误码
Content	String	是	关键词
CreatedAt	Integer	是	创建时间戳
EvilType	Integer	是	恶意类型 100：正常 20001：政治 20002：色情 20006：涉毒违法 20007：谩骂 20105：广告引流 24001：暴恐
Id	String	是	唯一标识
Label	Integer	是	样本类型 1：黑库 2：白库
Status	Integer	是	任务状态 1：已完成 2：处理中

User

用户相关信息

被如下接口引用：ManualReview, TextModeration。

名称	类型	必选	描述
AccountType	Integer	否	账号类别，"1-微信uin 2-QQ号 3-微信群uin 4-qq群号 5-微信openid 6-QQopenid 7-其它 string"

名称	类型	必选	描述
Age	Integer	否	年龄 默认0 未知
Gender	Integer	否	性别 默认0 未知 1 男性 2 女性
Level	Integer	否	用户等级，默认0 未知 1 低 2 中 3 高
Nickname	String	否	用户昵称
Phone	String	否	手机号
UserId	String	否	用户账号ID，如填写，会根据账号历史恶意情况，判定消息有害结果，特别是有利于可疑恶意情况下的辅助判断。账号可以填写微信uin、QQ号、微信openid、QQopenid、字符串等。该字段和账号类别确定唯一账号。

错误码

最近更新时间：2020-07-24 08:07:53

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
UnsupportedOperation	操作不支持。
ResourceInUse	资源被占用。
InternalError	内部错误。
RequestLimitExceeded	请求的次数超过了频率限制。
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。
LimitExceeded	超过配额限制。
NoSuchVersion	接口版本不存在。
ResourceNotFound	资源不存在。
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.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。
UnsupportedRegion	接口不支持所传地域。
UnauthorizedOperation	未授权操作。

错误码	说明
InvalidParameter	参数错误。
ResourceUnavailable	资源不可用。
AuthFailure.MFAFailure	MFA 错误。
AuthFailure.UnauthorizedOperation	请求未授权。请参考 CAM 文档对鉴权的说明。
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）。
AuthFailure.TokenFailure	token 错误。
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。
UnknownParameter	未知参数错误。
UnsupportedProtocol	HTTP(S)请求协议错误，只支持 GET 和 POST 请求。
InvalidParameterValue	参数取值错误。
InvalidAction	接口不存在。
MissingParameter	缺少参数错误。
ResourceInsufficient	资源不足。

业务错误码

错误码	说明
InvalidParameter.ParameterError	参数不可用
InvalidParameterValue.CustomAppId	渠道AppId参数错误
InvalidParameterValue.DuplicateContentID	重复提交ContentID
InvalidParameterValue.ErrAction	Action参数值错误
InvalidParameterValue.ErrAppId	AppId参数值错误
InvalidParameterValue.ErrFileContent	FileContent不可用：需Base64编码
InvalidParameterValue.ErrRequestID	RequestID参数值错误
InvalidParameterValue.ErrRequestSource	RequestSource请求参数错误或请求失败
InvalidParameterValue.ErrTextContentType	文本内容类型错误：需base64编码
InvalidParameterValue.ErrUin	Uin参数值错误
InvalidParameterValue.InvalidBatchId	BatchId取值错误
InvalidParameterValue.InvalidContent	Content参数错误

错误码	说明
InvalidParameterValue.InvalidContentID	ContentID参数错误
InvalidParameterValue.InvalidContentType	ContentType取值错误 1 图片 2 视频 3 文本 4 音频
InvalidParameterValue.InvalidCustomAppId	渠道AppId参数错误
InvalidParameterValue.InvalidImageContent	图片内容错误
InvalidParameterValue.InvalidParameter	参数取值错误
InvalidParameterValue.InvalidPriority	Priority取值错误
InvalidParameterValue.InvalidTitle	Title取值错误
MissingParameter.ErrFileUrl	FileUrl或FileContent都为空
ResourceNotFound.ErrDowdownInternalError	文件链接下载内部错误
ResourceNotFound.ErrDowdownParamsError	文件链接下载服务参数异常
ResourceNotFound.ErrDowdownSourceError	文件链接下载源错误
ResourceNotFound.ErrDowdownTimeOut	文件链接下载超时
ResourceUnavailable.ErrImageTimeOut	图片识别服务超时
ResourceUnavailable.ErrTextTimeOut	文本识别服务超时
UnauthorizedOperation.ErrAuth	鉴权失败
UnauthorizedOperation.Unauthorized	未开通权限