智能语音

API 文档

产品文档

腾讯云

文档目录

API 文档
更新历史

最近更新时间：2019-06-20 16:26:49

第 9 次发布

发布时间：2019-06-20 16:21:46

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- TextToVoice
  - 新增入参：Codec

第 8 次发布

发布时间：2018-08-30 15:40:25

本次发布包含了以下内容：

改善已有的文档。

第 7 次发布

发布时间：2018-08-23 22:15:03

本次发布包含了以下内容：

改善已有的文档。

第 6 次发布

发布时间：2018-08-17 18:31:37

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- Chat
- SimultaneousInterpreting

第 5 次发布

发布时间：2018-07-26 12:16:45

本次发布包含了以下内容：

改善已有的文档。

第 4 次发布

发布时间：2018-07-12 15:29:35

本次发布包含了以下内容：

改善已有的文档。

第 3 次发布

发布时间：2018-07-05 14:31:43

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- TextToVoice
  - **修改入参：**ProjectId

第 2 次发布

发布时间：2018-06-28 15:15:08

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- TextToVoice

第 1 次发布

发布时间：2018-05-24 17:04:19

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- SentenceRecognition

简介

最近更新时间：2019-06-20 16:26:49

智能语音服务 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。

智能语音服务（Artificial Audio Intelligence）满足语音识别、语音合成、声纹识别等语音处理需求。智能语音服务拥有强大的垂直领域定制化服务，打造专业高效的语音大脑，为企业提供全方位的智能语音解决方案。

腾讯语音识别(Automatic Speech Recognition, ASR) 为开发者提供语音转文字服务的最佳体验。

一句话识别(SentenceRecognition)可对60秒之内的短音频流进行识别。

API 概览

最近更新时间：2019-04-04 21:02:02

语音合成接口

接口名称	接口功能
TextToVoice	语音合成

语音识别相关接口

接口名称	接口功能
SentenceRecognition	一句话识别

其他接口

接口名称	接口功能
Chat	智能闲聊
SimultaneousInterpreting	同传

调用方式
请求结构

最近更新时间：2019-08-14 19:16:33

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 aai.tencentcloudapi.com ，也支持指定地域域名访问，例如广州地域的域名为 aai.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 aai.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。

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）	aai.tencentcloudapi.com
华南地区(广州)	aai.ap-guangzhou.tencentcloudapi.com
华东地区(上海)	aai.ap-shanghai.tencentcloudapi.com
华北地区(北京)	aai.ap-beijing.tencentcloudapi.com
西南地区(成都)	aai.ap-chengdu.tencentcloudapi.com
西南地区(重庆)	aai.ap-chongqing.tencentcloudapi.com
港澳台地区(中国香港)	aai.ap-hongkong.tencentcloudapi.com
亚太东南(新加坡)	aai.ap-singapore.tencentcloudapi.com
亚太东南(曼谷)	aai.ap-bangkok.tencentcloudapi.com
亚太南部(孟买)	aai.ap-mumbai.tencentcloudapi.com
亚太东北(首尔)	aai.ap-seoul.tencentcloudapi.com
亚太东北(东京)	aai.ap-tokyo.tencentcloudapi.com
美国东部(弗吉尼亚)	aai.na-ashburn.tencentcloudapi.com
美国西部(硅谷)	aai.na-siliconvalley.tencentcloudapi.com
北美地区(多伦多)	aai.na-toronto.tencentcloudapi.com
欧洲地区(法兰克福)	aai.eu-frankfurt.tencentcloudapi.com
欧洲地区(莫斯科)	aai.eu-moscow.tencentcloudapi.com

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

金融区接入地域	金融区域名
华东地区(上海金融)	aai.ap-shanghai-fsi.tencentcloudapi.com
华南地区(深圳金融)	aai.ap-shenzhen-fsi.tencentcloudapi.com

2. 通信协议

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

3. 请求方法

支持的 HTTP 请求方法:

- POST（推荐）
- GET

POST 请求支持的 Content-Type 类型：

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

GET 请求的请求包大小不得超过32KB。POST 请求使用签名方法为 HmacSHA1、HmacSHA256 时不得超过1MB。POST 请求使用签名方法为 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：2019-08-14 19:16:33

公共参数是用于标识用户和接口鉴权目的的参数，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

使用 TC3-HMAC-SHA256 签名方法时，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	是	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入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
Authorization	String	是	HTTP 标准身份认证头部字段，例如： TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024 其中， - TC3-HMAC-SHA256：签名方法，目前固定取该值； - Credential：签名凭证，AKIDEXAMPLE 是 SecretId；Date 是 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致；service 为产品名，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 aai； - 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。
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=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE/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=AKIDEXAMPLE/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=AKIDEXAMPLE/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 HmacSHA1 和 HmacSHA256 签名方法时，公共参数需要统一放到请求串中，如下

参数名称	类型	必选	描述

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	是	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。具体计算方法参见接口鉴权文档。
Version	String	是	操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
SignatureMethod	String	否	签名方式，目前支持 HmacSHA256 和 HmacSHA1。只有指定此参数为 HmacSHA256 时，才使用 HmacSHA256 算法验证签名，其他情况均使用 HmacSHA1 验证签名。
Token	String	否	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=AKIDEXAMPLE

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=AKIDEXAMPLE
```

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。

区域	取值
亚太东南(曼谷)	ap-bangkok
华北地区(北京)	ap-beijing
西南地区(成都)	ap-chengdu
西南地区(重庆)	ap-chongqing
华南地区(广州)	ap-guangzhou
华南地区(广州Open)	ap-guangzhou-open
港澳台地区(中国香港)	ap-hongkong
亚太南部(孟买)	ap-mumbai
亚太东北(首尔)	ap-seoul
华东地区(上海)	ap-shanghai
华东地区(上海金融)	ap-shanghai-fsi
华南地区(深圳金融)	ap-shenzhen-fsi
亚太东南(新加坡)	ap-singapore
欧洲地区(法兰克福)	eu-frankfurt
欧洲地区(莫斯科)	eu-moscow
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

接口鉴权 v3

最近更新时间：2019-08-14 19:16:33

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。如已泄漏，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】即可以创建一对密钥。

使用开发者资源

腾讯云 API 配套了 7 种常见的编程语言 SDK，均已开源，包括 Python、Java、PHP、Go、NodeJS、.NET、C++。同时还提供了 API Explorer 供用户在线调用、签名验证、生成 SDK 代码。如果在签名时遇到困难，请善加利用这些资源。

TC3-HMAC-SHA256 签名方法

TC3-HMAC-SHA256 签名方法相比以前的 HmacSHA1 和 HmacSHA256 签名方法，功能上覆盖了以前的签名方法，而且更安全，支持更大的请求，支持 json 格式，性能有一定提升，建议使用该签名方法计算签名。

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

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

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

在示例中，不论公共参数或者接口的参数，我们尽量选择容易犯错的情况。在实际调用接口时，请根据实际情况来，每个接口的参数并不相同，不要照抄这个例子的参数和值。

假设用户的 SecretId 和 SecretKey 分别是：AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE 和 Gu5t9xGARNpq86cd98joQYCN3EXAMPLE。用户想查看广州区云服务器名为"未命名"的主机状态，只返回一条数据。则请求可能为：

```
curl -X POST https://cvm.tencentcloudapi.com \
-H "Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host, Signature=72e494ea809ad7a8c8f7a4507b9bddcbaa8e581f516e8da2f66e2c5a96525168" \
-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 需要经过 URL 编码。
CanonicalHeaders	参与签名的头部信息，至少包含 host 和 content-type 两个头部，也可加入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 Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))。此示例计算结果是 `5ffe6a04c0664d6b969fab9a13bdab201d63ee709638e2749d62a09ca18d7031` 。

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 = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"
SecretDate = HMAC_SHA256("TC3" + SecretKey, Date)
SecretService = HMAC_SHA256(SecretDate, Service)
SecretSigning = HMAC_SHA256(SecretService, "tc3_request")
```

字段名称	解释
SecretKey	原始的 SecretKey，即 `Gu5t9xGARNpq86cd98joQYCN3EXAMPLE` 。
Date	即 Credential 中的 Date 字段信息。此示例取值为 `2019-02-25` 。
Service	即 Credential 中的 Service 字段信息。此示例取值为 `cvm` 。

2）计算签名，伪代码如下：

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

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 AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE 。
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 72e494ea809ad7a8c8f7a4507b9bddcbaa8e581f516e8da2f66e2c5a96525168 。

根据以上规则，示例中得到的值为：

```
TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host, Signature=72e494ea809ad7a8c8f7a4507b9bddcbaa8e581f516e8da2f66e2c5a96525168
```

最终完整的调用信息如下：

```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host, Signature=72e494ea809ad7a8c8f7a4507b9bddcbaa8e581f516e8da2f66e2c5a96525168
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"}]}
```

5. 签名演示

Java

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

public class TencentCloudAPITC3Demo {
```

```
private final static Charset UTF8 = StandardCharsets.UTF_8;
private final static String SECRET_ID = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE";
private final static String SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE";
private final static String CT_JSON = "application/json; charset=utf-8";

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

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

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

// ************ 步骤 1：拼接规范请求串 ************
String httpRequestMethod = "POST";
String canonicalUri = "/";
String canonicalQueryString = "";
String canonicalHeaders = "content-type:application/json; charset=utf-8\n" + "host:" + host + "\n";
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 = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"

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

签名失败

存在以下签名失败的错误码，请根据实际情况处理。

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

接口鉴权

最近更新时间：2019-08-14 19:16:33

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】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: AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE
- SecretKey: Gu5t9xGARNpq86cd98joQYCN3EXAMPLE

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	AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE
Timestamp	当前时间戳	1465185768
Nonce	随机正整数	11886
Region	实例所在区域	ap-guangzhou
InstanceIds.0	待查询的实例 ID	ins-09dx96dg

参数名称	中文	参数值
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',
'InstanceIds.0' : 'ins-09dx96dg',
'Limit' : 20,
'Nonce' : 11886,
'Offset' : 0,
'Region' : 'ap-guangzhou',
'SecretId' : 'AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE',
'Timestamp' : 1465185768,
'Version': '2017-03-12',
}
```

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

2.2. 拼接请求字符串

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

然后将格式化后的各个参数用"&"拼接在一起，最终生成的请求字符串为:

```
Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Timestamp=1465185768&Version=2017-03-12
```

2.3. 拼接签名原文字符串

此步骤生成签名原文字符串。 签名原文字符串由以下几个参数构成:

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

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

示例的拼接结果为：

```
GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&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 = 'Gu5t9xGARNpq86cd98joQYCN3EXAMPLE';
$srcStr = 'GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&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：

```
EliP9YW3pW28FpsEdkXt/+WcGeI=
```

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 EliP9YW3pW28FpsEdkXt/+WcGeI= ，最终得到的签名串请求参数（Signature）为：EliP9YW3pW28FpsEdkXt/+WcGeI=，它将用于生成最终的请求 URL。

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

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

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

4. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理。

错误代码	错误描述
AuthFailure.SignatureExpire	签名过期
AuthFailure.SecretIdNotFound	密钥不存在

错误代码	错误描述
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
- JavaScript
- .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&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Signature=EliP9YW3pW28FpsEdkXt/+WcGeI=&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.DatatypeConverter;

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

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

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

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

public static void main(String[] args) throws Exception {
TreeMap<String, Object> params = new TreeMap<String, Object>(); // TreeMap可以自动排序
// 实际调用时应当使用随机数，例如：params.put("Nonce", new Random().nextInt(java.lang.Integer.MAX_VALUE));
params.put("Nonce", 11886); // 公共参数
// 实际调用时应当使用系统当前时间，例如： params.put("Timestamp", System.currentTimeMillis() / 1000);
params.put("Timestamp", 1465185768); // 公共参数
params.put("SecretId", "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"); // 公共参数
params.put("Action", "DescribeInstances"); // 公共参数
params.put("Version", "2017-03-12"); // 公共参数
params.put("Region", "ap-guangzhou"); // 公共参数
params.put("Limit", 20); // 业务参数
params.put("Offset", 0); // 业务参数
params.put("InstanceIds.0", "ins-09dx96dg"); // 业务参数
params.put("Signature", sign(getStringToSign(params), "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE", "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 = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"

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

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

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

返回结果

最近更新时间：2019-08-14 19:16:33

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。
UnsupportedRegion	接口不支持所传地域。

语音合成接口
语音合成

最近更新时间：2019-08-08 23:15:43

1. 接口描述

接口请求域名： aai.tencentcloudapi.com 。

腾讯云语音合成技术（TTS）可以将任意文本转化为语音，实现让机器和应用张口说话。 腾讯TTS技术可以应用到很多场景，比如，移动APP语音播报新闻；智能设备语音提醒；依靠网上现有节目或少量录音，快速合成明星语音，降低邀约成本；支持车载导航语音合成的个性化语音播报。 内测期间免费使用。

默认接口请求频率限制：20次/秒。

注意：本接口支持金融区地域。由于金融区和非金融区是隔离不互通的，因此当公共参数 Region 为金融区地域（例如 ap-shanghai-fsi）时，需要同时指定带金融区地域的域名，最好和 Region 的地域保持一致，例如：aai.ap-shanghai-fsi.tencentcloudapi.com。

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：TextToVoice
Version	是	String	公共参数，本接口取值：2018-05-22
Region	是	String	公共参数，详见产品支持的 地域列表。
Text	是	String	合成语音的源文本，按UTF-8编码统一计算。 中文最大支持100个汉字（全角标点符号算一个汉字）；英文最大支持400个字母（半角标点符号算一个字母）。包含空格等字符时需要url encode再传输。
SessionId	是	String	一次请求对应一个SessionId，会原样返回，建议传入类似于uuid的字符串防止重复。
ModelType	是	Integer	模型类型，1-默认模型。
Volume	否	Float	音量大小，范围：[0，10]，分别对应11个等级的音量，默认为0，代表正常音量。没有静音选项。 输入除以上整数之外的其他参数不生效，按默认值处理。
Speed	否	Float	语速，范围：[-2，2]，分别对应不同语速： • -2代表0.6倍 • -1代表0.8倍 • 0代表1.0倍（默认） • 1代表1.2倍 • 2代表1.5倍 输入除以上整数之外的其他参数不生效，按默认值处理。
ProjectId	否	Integer	项目id，用户自定义，默认为0。

参数名称	必选	类型	描述
VoiceType	否	Integer	音色 • 0-亲和女声(默认) • 1-亲和男声 • 2-成熟男声 • 3-活力男声 • 4-温暖女声 • 5-情感女声 • 6-情感男声
PrimaryLanguage	否	Integer	主语言类型： • 1-中文（默认） • 2-英文
SampleRate	否	Integer	音频采样率： • 16000：16k（默认） • 8000：8k
Codec	否	String	返回音频格式，可取值：wav（默认），mp3

3. 输出参数

参数名称	类型	描述
Audio	String	base64编码的wav/mp3音频数据
SessionId	String	一次请求对应一个SessionId
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 API调用

输入示例

```
https://aai.tencentcloudapi.com/?Action=TextToVoice
&Text=你好
&SessionId=session-1234
&Volume=1
&Speed=1
&ProjectId=0
&ModelType=1
&PrimaryLanguage=1
&SampleRate=16000
&Codec=wav
&<公共请求参数>
```

输出示例


```
{
"Response": {
"Audio": "UklGRlR/AABXQVZFZm10IBAAAAABAAEAgD4AAAB9AAACABAAZGF0YSx9AAD+////AQD//wAAAAAAAAIAAQADAAMABgAEAAYABQAGAAUABwAIAAgACQAE.....AAgACAAEAAgADAAIAAwACAAQAAwACAAIAAgADAAMAAgACAAIAAwABAAAAAAAAAAAAAAD/////AAAAAAAA//8AAP///v/9//7//v///////v8AAP///////wAA/////wAA/////wAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA",
"RequestId": "9a7a1615-3e09-4db2-8032-5c6f497f7e6a",
"SessionId": "session-1234"
}
}
```


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
InternalError	内部错误
InternalError.TextConvertFailed	文本转换失败，请检查文本格式。
InvalidParameter.ErrorNoBodydata	没有body数据。
InvalidParameterValue	参数取值错误
InvalidParameterValue.AppIdNotRegistered	appid未注册。

错误码	描述
UnsupportedOperation.TextTooLong	文本过长，请参考请求参数Text的说明。

错误码

最近更新时间：2019-08-08 19:13:47

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
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）。
AuthFailure.MFAFailure	MFA 错误。
AuthFailure.SecretIdNotFound	密钥不存在。 请在控制台检查密钥是否已被删除或者禁用，如状态正常，请检查密钥是否填写正确，注意前后不得有空格。
AuthFailure.SignatureExpire	签名过期。Timestamp 和服务器时间相差不得超过五分钟，请检查本地时间是否和标准时间同步。
AuthFailure.SignatureFailure	签名错误。 签名计算错误，请对照调用方式中的接口鉴权文档检查签名计算过程。
AuthFailure.TokenFailure	token 错误。
AuthFailure.UnauthorizedOperation	请求未授权。请参考 CAM 文档对鉴权的说明。
DryRunOperation	DryRun 操作，代表请求将会是成功的，只是多传了 DryRun 参数。
FailedOperation	操作失败。
InternalError	内部错误。
InvalidAction	接口不存在。
InvalidParameter	参数错误。

错误码	说明
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	HTTP(S)请求协议错误，只支持 GET 和 POST 请求。
UnsupportedRegion	接口不支持所传地域。

业务错误码

错误码	说明
FailedOperation.ServiceIsolate	账号因为欠费停止服务，请在腾讯云账户充值。
FailedOperation.UserHasNoFreeAmount	账号本月免费额度已用完。
FailedOperation.UserNotRegistered	服务未开通，请在腾讯云官网语音识别控制台开通服务。
InternalError	内部错误
InternalError.ErrorConfigure	初始化配置失败。
InternalError.ErrorCreateLog	创建日志失败。
InternalError.ErrorDownFile	下载音频文件失败。
InternalError.ErrorFailNewprequest	新建数组失败。
InternalError.ErrorFailWritetodb	写入数据库失败。
InternalError.ErrorFileCannotopen	文件无法打开。
InternalError.ErrorGetRoute	获取路由失败。
InternalError.ErrorMakeLogpath	创建日志路径失败。

错误码	说明
InternalError.ErrorRecognize	识别失败。
InternalError.ErrorTranslate	机器翻译失败。
InternalError.TextConvertFailed	文本转换失败，请检查文本格式。
InvalidParameter.ErrorChatText	错误的聊天输入文本。
InvalidParameter.ErrorChatUser	错误的User参数。
InvalidParameter.ErrorContentlength	请求数据长度无效。
InvalidParameter.ErrorNoBodydata	没有body数据。
InvalidParameter.ErrorParamsMissing	参数不全。
InvalidParameter.ErrorParsequest	解析请求数据失败。
InvalidParameterValue	参数取值错误
InvalidParameterValue.AppIdNotRegistered	appid未注册。
InvalidParameterValue.ErrorInvalidAppid	AppId无效。
InvalidParameterValue.ErrorInvalidClientip	ClientIp无效。
InvalidParameterValue.ErrorInvalidEngservice	EngSerViceType无效。
InvalidParameterValue.ErrorInvalidOpenTranslate	是否进行翻译。
InvalidParameterValue.ErrorInvalidProjectid	ProjectId无效。
InvalidParameterValue.ErrorInvalidRequestid	RequestId无效。
InvalidParameterValue.ErrorInvalidSourceLanguage	翻译的源语言类型不支持。
InvalidParameterValue.ErrorInvalidSourcetype	SourceType无效。
InvalidParameterValue.ErrorInvalidSubservicetype	SubserviceType无效。
InvalidParameterValue.ErrorInvalidTargetLanguage	翻译的目标语言类型不支持。
InvalidParameterValue.ErrorInvalidUrl	Url无效。
InvalidParameterValue.ErrorInvalidUseraudiokey	UsrAudioKey无效。
InvalidParameterValue.ErrorInvalidVoiceFormat	音频编码格式不支持。
InvalidParameterValue.ErrorInvalidVoicedata	音频数据无效。
UnsupportedOperation.TextTooLong	文本过长，请参考请求参数Text的说明。

智能语音服务 API 2017
语音合成

最近更新时间：2019-08-27 17:22:56

1. 接口描述

接口请求域名： aai.cloud.tencent.com/tts
接口请求频率限制：20次/每秒。
腾讯云语音合成技术（TTS）可以将任意文本转化为语音，实现让机器和应用张口说话。 腾讯 TTS 技术可以应用到很多场景，例如，移动 App 语音播报新闻；智能设备语音提醒；依靠网上现有节目或少量录音，快速合成明星语音，降低邀约成本；支持车载导航语音合成的个性化语音播报。**本接口内测期间免费使用。**

2. 输入参数

注意：
请求的 JSON 参数需要按照鉴权的顺序排序。

参数名称	必选	类型	描述
Action	是	String	本接口取值：TextToStreamAudio
AppId	是	Integer	账号 AppId
SecretId	是	String	官网 SecretId
Timestamp	是	Integer	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。
Expired	是	Integer	签名的有效期，是一个符合 UNIX Epoch 时间戳规范的数值，单位为秒；Expired 必须大于 Timestamp 且 Expired-Timestamp 小于90天。
Text	是	String	合成语音的源文本。中文最大支持600个汉字（全角标点符号算一个汉字）；英文最大支持1800个字母（半角标点符号算一个字母）。包含空格等字符时需要 URL encode 再传输。
SessionId	是	String	一次请求对应一个 SessionId，会原样返回，建议传入类似于 uuid 的字符串防止重复。
ModelType	否	Integer	模型类型，1：默认模型
Volume	否	Float	音量大小，范围：[0，10]，分别对应11个等级的音量，默认值为0，代表正常音量。没有静音选项。 输入除以上整数之外的其他参数不生效，按默认值处理。

参数名称	必选	类型	描述
Speed	否	Integer	语速，范围：[-2，2]，分别对应不同语速： -2代表0.6倍 -1代表0.8倍 0代表1.0倍（默认） 1代表1.2倍 2代表1.5倍 输入除以上整数之外的其他参数不生效，按默认值处理。
ProjectId	否	Integer	项目 ID，用户自定义，默认为0。
VoiceType	否	Integer	音色： 0：亲和女声（默认） 1：亲和男声 2：成熟男声 3：活力男声 4：温暖女声 5：情感女声 6：情感男声
PrimaryLanguage	否	Integer	主语言类型： 1：中文（默认） 2：英文
SampleRate	否	Integer	音频采样率： 16000：16k（默认） 8000：8k
Codec	否	String	返回音频格式： opus：返回多段含 opus 压缩分片音频，数据量小，建议使用（默认）。 pcm：返回二进制 pcm 音频，使用简单，但数据量大。

请求 headers

```
{
"Content-Type":"application/json",
"Authorization":"HRCKlbwPhWtVvfGn914qE5O1rwc="
}
```

鉴权说明请参考 接口鉴权 文档。

注：仅需要生成签名串，签名串不需要编码。

3. 输出参数

协议说明：使用 HTTP Chunk 协议，一次 HTTP 请求内按序返回多段分片直到音频结束。语音的输出不带长度和时间信息。

Opus

Codec 参数为 opus 返回，默认返回。返回多个语音分片，分片大小不一，单个分片格式说明如下：

标识头“opus”H (4字节字符串)	分片序号 S (4字节二进制数据)	分片音频长度 L (4字节二进制数据)	分片音频 D (长度为分片音频长度 L)
标识新的分片的开始	序号从0开始，-1结束	按分片音频长度读取音频数据	base64后的 Opus 压缩音频

其中最后一片音频（序号 S = -1）数据固定为“AAAA”，该段数据无效。

Pcm

Codec 参数为 pcm 返回，同等条件下返回数据量约为 Opus 格式的 10 倍。
返回格式：二进制
返回内容：pcm 音频流

4. 示例

Opus

Codec 参数为 opus 返回，Java 示例如下：

```
//填写请求参数
mRequestBody = "{"+
"\"AppId\":1255824371,"+
"\"SecretId\":\"AKID3wb2D8EFRN3UgbwengJ5h5E8nERPabqq\","+
"\"PrimaryLanguage\": 1,"+
"\"SampleRate\": 16000,"+
"\"Action\":\"TextToStreamAudio\","+
"\"VoiceType\":1,"+
"\"Text\":\"我只是拿来测试的文本\","+
"\"SessionId\":\"session-1234\","+
"\"Timestamp\":1535362116,"+
"\"Expired\":1535365716,"+
"\"Speed\":0,"+
"\"Volume\":0"+
"}";

//http post请求
URL url = new URL(SERVER_URL);
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod("POST");
conn.setRequestProperty("Content-Type", "application/json");
conn.setRequestProperty("Authorization", "VVerlEkz2OAuSBdNtH6W2M1xal0=");
conn.connect();
OutputStream out = conn.getOutputStream();
out.write(postJsonBody.getBytes("UTF-8"));
out.flush();
out.close();
InputStream inputStream = conn.getInputStream();

//解析返回数据
while (!Thread.currentThread().isInterrupted()) {
//read header
byte[] headBuffer = new byte[4];
```

```
fill(inputStream, headBuffer);

//read seq
byte[] seqBuffer = new byte[4];
fill(inputStream, seqBuffer);
int seq = bytesToInt(seqBuffer);

//read pkg size
byte[] pbPkgSizeHeader = new byte[4];
fill(inputStream, pbPkgSizeHeader);
int pbPkgSize = bytesToInt(pbPkgSizeHeader);

//read pkg
byte[] pbPkg = new byte[pbPkgSize];
fill(inputStream, pbPkg);

//init decoder
if (decoder == null) {
decoder = new OpusDecoder();
}
//decode
short[] pcm = decoder.decodeTTSData(seq, pbPkg);

//enqueue
//save pcm to a list or a buffer

//end
if (seq == -1) {
break;
}
}


private static boolean fill(InputStream in, byte[] buffer) throws IOException {
int length = buffer.length;
int hasRead = 0;
while (true) {
int offset = hasRead;
int count = length - hasRead;
int currentRead = in.read(buffer, offset, count);
if (currentRead >= 0) {
hasRead += currentRead;
if (hasRead == length) {
return true;
}
}
if (currentRead == -1) {
return false;
}
}
}
```

Pcm

Codec 参数为 pcm 返回，Python 示例如下：

```
def tts_wav_demo():
header = {
"Content-Type": "application/json",
"Authorization": "VVerIEkz2OAuSBdNtH6W2M1xal0="
}
request_data = {
"AppId": 1255824371,
"SecretId": "AKID3wb2D8EFRN3UgbwengJ5h5E8nERPabqq",
"PrimaryLanguage": 1,
"SampleRate": 16000,
"Action": "TextToStreamAudio",
"VoiceType": 1,
"Text": "我只是拿来测试的文本",
"SessionId": "session-1234",
"Timestamp": 1535362116,
"Expired": 1535365716,
"Speed": 0,
"Volume": 0
}
r = requests.post(req_url, headers=header, data=request_data)
file_object = open("./test.pcm", "w")
for chunk in r.iter_content(1000):
file_object.write(chunk)
file_object.close()
```

Codec 参数为 pcm 返回，Java 示例如下：

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.io.*;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.*;

public class TtsController {
private static final String SERVER_URL = "https://aai.cloud.tencent.com/tts";
private static final String DOMAIN_NAME ="aai.cloud.tencent.com/tts";
private static final String SECRET_KEY = "********************************";
private static final String SECRET_ID = "********************************";

public static void main(String[] args){
TreeMap<String, String> requestMap = new TreeMap<>();
requestMap.put("Action","TextToStreamAudio");
requestMap.put("Codec", "pcm");
requestMap.put("Text", "我只是拿来测试的文本");
requestMap.put("SessionId", "session-1234");
requestMap.put("AppId", "1255824371");
```

```
requestMap.put("Timestamp", ""+ System.currentTimeMillis()/1000L);
requestMap.put("Expired", ""+ (System.currentTimeMillis()/1000L + 600));
requestMap.put("SecretId", SECRET_ID);
new WorkingThread(requestMap).start();
}

private static final class WorkingThread extends Thread {

private TreeMap<String, String> requestMap;
private String mRequestBody;

WorkingThread(TreeMap<String, String> requestMap) {
this.requestMap = requestMap;
mRequestBody = "{\n" +
"\t\"Action\":\""+ requestMap.get("Action") +"\",\n" +
"\t\"Codec\":\""+ requestMap.get("Codec")+"\",\n" +
"\t\"Text\":\"" + requestMap.get("Text") + "\",\n" +
"\t\"SessionId\":\""+requestMap.get("SessionId")+"\",\n" +
"\t\"Timestamp\":"+requestMap.get("Timestamp")+",\n" +
"\t\"Expired\":"+requestMap.get("Expired")+",\n" +
"\t\"SecretId\":\""+requestMap.get("SecretId")+"\",\n" +
"\t\"AppId\":"+requestMap.get("AppId") +"\n" +
"}";
System.out.println(mRequestBody);
}

@Override
public void run() {
InputStream inputStream = null;
try {
inputStream = obtainResponseStreamWithJava(mRequestBody, requestMap);
processProtocolBufferStream(inputStream);
} catch (IOException e) {
e.printStackTrace();
}
}

private void processProtocolBufferStream(final InputStream inputStream) throws FileNotFoundException {
FileOutputStream fos = new FileOutputStream("test.pcm",true);
boolean fillSuccess;

while (true) {
byte[] pcmData = new byte[1024];
try {
fillSuccess = fill(inputStream, pcmData);
fos.write(pcmData);
if(!fillSuccess){
fos.flush();
fos.close();
break;
}
} catch (IOException e) {
e.printStackTrace();
}
```

```
}
}

}

private static InputStream obtainResponseStreamWithJava(String postJsonBody, TreeMap<String,String> requestMap) t
hrows IOException {
//发送POST请求
URL url = new URL(SERVER_URL);
HttpURLConnection conn = (HttpURLConnection)url.openConnection();
String authorization = generateSign(requestMap);
conn.setRequestMethod("POST");
conn.setDoOutput(true);
conn.setRequestProperty("Content-Type", "application/json");
conn.setRequestProperty("Authorization", authorization);
conn.connect();
OutputStream out = conn.getOutputStream();
out.write(postJsonBody.getBytes(StandardCharsets.UTF_8));
out.flush();
out.close();

return conn.getInputStream();
}

private static String generateSign(TreeMap<String, String> params) {
String paramStr = "POST"+ DOMAIN_NAME + "?";
StringBuilder builder = new StringBuilder(paramStr);
for (Map.Entry<String, String> entry: params.entrySet()) {
builder.append(String.format(Locale.CHINESE,"%s=%s", entry.getKey(), String.valueOf(entry.getValue())))
.append("&");
}

//去掉最后一个&
builder.deleteCharAt(builder.lastIndexOf("&"));

String sign = "";
String source = builder.toString();
System.out.println(source);
Mac mac = null;
try {
mac = Mac.getInstance("HmacSHA1");
SecretKeySpec keySpec = new SecretKeySpec(SECRET_KEY.getBytes(), "HmacSHA1");
mac.init(keySpec);
mac.update(source.getBytes());
sign = Base64.getEncoder().encodeToString(mac.doFinal());
} catch (NoSuchAlgorithmException | InvalidKeyException e) {
e.printStackTrace();
}

System.out.println("生成签名串：" + sign);
return sign;
}

/**
```

```
 * 从 InputStream 读取内容到 buffer, 直到 buffer 填满
 * @return 如果 InputStream 内容不足以填满 buffer, 则返回 false.
 * @throws IOException 可能抛出的异常
 */
private static boolean fill(InputStream in, byte[] buffer) throws IOException {
int length = buffer.length;
int hasRead = 0;
while (true) {
int offset = hasRead;
int count = length - hasRead;
int currentRead = in.read(buffer, offset, count);
if (currentRead >= 0) {
hasRead += currentRead;
if (hasRead == length) {
return true;
}
}
if (currentRead == -1) {
return false;
}
}
}

}
```