

消息队列 RabbitMQ 版 API 文档（Serverless 版）



腾讯云

【版权声明】

©2013–2025 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或95716。

文档目录

API 文档（Serverless 版）

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

参数类型

RabbitMQ Serverless 实例管理相关接口

修改RabbitMQ vhost信息

修改RabbitMQ的用户

修改RabbitMQ队列

修改RabbitMQ的权限

修改RabbitMQ exchange

查询RabbitMQ vhost列表

查询RabbitMQ用户列表

查询RabbitMQ队列列表

查询RabbitMQ队列详情

查询RabbitMQ权限列表

查询RabbitMQ exchange 列表

查询RabbitMQ exchange 详情

查询RabbitMQ队列消费者列表

查询RabbitMQ连接列表

查询RabbitMQ路由关系列表

删除RabbitMQ的vhost

删除RabbitMQ的用户

删除RabbitMQ队列

删除RabbitMQ的权限

删除RabbitMQ exchange

解绑RabbitMQ路由关系

创建RabbitMQ的vhost

创建RabbitMQ的用户

创建RabbitMQ队列

创建RabbitMQ exchange

创建RabbitMQ路由关系

RabbitMQ Serverless PAAS 能力相关接口

修改 rabbitmq serverless 实例信息

获取 RabbitMQ Serverless 集群列表

获取 rabbitmq serverless 实例总体信息

数据结构

错误码

API 文档（Serverless 版）

更新历史

最近更新时间：2025-05-29 02:07:04

第 3 次发布

发布时间：2025-05-29 02:06:57

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- [RabbitMQClusterSpecInfo](#)
 - 新增成员：Features
- [RabbitMQServerlessInstance](#)
 - 新增成员：ServerlessExt

第 2 次发布

发布时间：2025-02-20 02:13:01

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- [RabbitMQServerlessInstance](#)
 - 新增成员：IsolatedTime

第 1 次发布

发布时间：2025-02-17 15:13:11

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- [CreateRabbitMQServerlessBinding](#)
- [CreateRabbitMQServerlessExchange](#)
- [CreateRabbitMQServerlessQueue](#)
- [CreateRabbitMQServerlessUser](#)
- [CreateRabbitMQServerlessVirtualHost](#)
- [DeleteRabbitMQServerlessBinding](#)
- [DeleteRabbitMQServerlessExchange](#)
- [DeleteRabbitMQServerlessPermission](#)
- [DeleteRabbitMQServerlessQueue](#)
- [DeleteRabbitMQServerlessUser](#)
- [DeleteRabbitMQServerlessVirtualHost](#)
- [DescribeRabbitMQServerlessBindings](#)
- [DescribeRabbitMQServerlessConnection](#)
- [DescribeRabbitMQServerlessConsumers](#)

- [DescribeRabbitMQServerlessExchangeDetail](#)
- [DescribeRabbitMQServerlessExchanges](#)
- [DescribeRabbitMQServerlessInstance](#)
- [DescribeRabbitMQServerlessPermission](#)
- [DescribeRabbitMQServerlessQueueDetail](#)
- [DescribeRabbitMQServerlessQueues](#)
- [DescribeRabbitMQServerlessUser](#)
- [DescribeRabbitMQServerlessVirtualHost](#)
- [ListRabbitMQServerlessInstances](#)
- [ModifyRabbitMQServerlessExchange](#)
- [ModifyRabbitMQServerlessInstance](#)
- [ModifyRabbitMQServerlessPermission](#)
- [ModifyRabbitMQServerlessQueue](#)
- [ModifyRabbitMQServerlessUser](#)
- [ModifyRabbitMQServerlessVirtualHost](#)

新增数据结构：

- [ExchangeQuota](#)
- [Filter](#)
- [QueueQuota](#)
- [RabbitMQBindingListInfo](#)
- [RabbitMQClusterInfo](#)
- [RabbitMQClusterSpecInfo](#)
- [RabbitMQConnection](#)
- [RabbitMQConsumersListInfo](#)
- [RabbitMQExchangeListInfo](#)
- [RabbitMQPermission](#)
- [RabbitMQQueueListConsumerDetailInfo](#)
- [RabbitMQQueueListInfo](#)
- [RabbitMQServerlessAccessInfo](#)
- [RabbitMQServerlessEndpoint](#)
- [RabbitMQServerlessInstance](#)
- [RabbitMQServerlessWhiteListInfo](#)
- [RabbitMQUser](#)
- [RabbitMQVirtualHostInfo](#)
- [RabbitMQVirtualHostStatistics](#)
- [UserQuota](#)
- [VirtualHostQuota](#)
- [VpcEndpointInfo](#)

简介

最近更新时间：2025-04-25 02:04:06

基于 RocketMQ 底座实现 AMQP 协议，提供跟 RabbitMQ 完全兼容的能力。

API 概览

最近更新时间：2025-02-17 15:13:47

RabbitMQ Serverless PAAS 能力相关接口

接口名称	接口功能	频率限制（次/秒）
DescribeRabbitMQServerlessInstance	获取 rabbitmq serverless 实例总体信息	20
ListRabbitMQServerlessInstances	获取 RabbitMQ Serverless 集群列表	20
ModifyRabbitMQServerlessInstance	修改 rabbitmq serverless 实例信息	20

RabbitMQ Serverless 实例管理相关接口

接口名称	接口功能	频率限制（次/秒）
CreateRabbitMQServerlessBinding	创建RabbitMQ路由关系	20
CreateRabbitMQServerlessExchange	创建RabbitMQ exchange	20
CreateRabbitMQServerlessQueue	创建RabbitMQ队列	20
CreateRabbitMQServerlessUser	创建RabbitMQ的用户	20
CreateRabbitMQServerlessVirtualHost	创建RabbitMQ的vhost	20
DeleteRabbitMQServerlessBinding	解绑RabbitMQ路由关系	20
DeleteRabbitMQServerlessExchange	删除RabbitMQ exchange	20
DeleteRabbitMQServerlessPermission	删除RabbitMQ的权限	20
DeleteRabbitMQServerlessQueue	删除RabbitMQ队列	20
DeleteRabbitMQServerlessUser	删除RabbitMQ的用户	20
DeleteRabbitMQServerlessVirtualHost	删除RabbitMQ的vhost	20
DescribeRabbitMQServerlessBindings	查询RabbitMQ路由关系列表	20
DescribeRabbitMQServerlessConnection	查询RabbitMQ连接列表	20
DescribeRabbitMQServerlessConsumers	查询RabbitMQ队列消费者列表	20
DescribeRabbitMQServerlessExchangeDetail	查询RabbitMQ exchange 详情	20
DescribeRabbitMQServerlessExchanges	查询RabbitMQ exchange 列表	20
DescribeRabbitMQServerlessPermission	查询RabbitMQ权限列表	20
DescribeRabbitMQServerlessQueueDetail	查询RabbitMQ队列详情	20
DescribeRabbitMQServerlessQueues	查询RabbitMQ队列表	20
DescribeRabbitMQServerlessUser	查询RabbitMQ用户列表	20
DescribeRabbitMQServerlessVirtualHost	查询RabbitMQ vhost列表	20
ModifyRabbitMQServerlessExchange	修改RabbitMQ exchange	20

接口名称	接口功能	频率限制（次/秒）
ModifyRabbitMQServerlessPermission	修改RabbitMQ的权限	20
ModifyRabbitMQServerlessQueue	修改RabbitMQ队列	20
ModifyRabbitMQServerlessUser	修改RabbitMQ的用户	20
ModifyRabbitMQServerlessVirtualHost	修改RabbitMQ vhost信息	20

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

调用方式

请求结构

最近更新时间：2025-05-21 01:33:52

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

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）	trabbit.ap-shanghai-fsi.tencentcloudapi.com
华南地区（深圳金融）	trabbit.ap-shenzhen-fsi.tencentcloudapi.com

2. 通信协议

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

3. 请求方法

支持的 HTTP 请求方法：

- POST（推荐）
- GET

POST 请求支持的 Content-Type 类型：

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

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

4. 字符编码

均使用 UTF-8 编码。

公共参数

最近更新时间：2025-02-21 02:04:51

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

公共参数的具体内容会因您使用的签名方法版本不同而有所差异。

使用签名方法 v3 的公共参数

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

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

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

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

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

HTTP GET 请求结构示例：

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

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

HTTP POST (application/json) 请求结构示例:

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

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

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

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

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

```
Authorization: TC3-HMAC-SHA256 Credential=AKID*****/2018-05-30/cvm/tc3_request,
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--
```

使用签名方法 v1 的公共参数

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

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

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

HTTP GET 请求结构示例：

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

```
Host: cvm.tencentcloudapi.com
```

HTTP POST 请求结构示例：

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

```
Host: cvm.tencentcloudapi.com
```

```
Content-Type: application/x-www-form-urlencoded
```

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

地域列表

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

地域	取值
华南地区（广州）	ap-guangzhou
亚太东北（首尔）	ap-seoul
华东地区（上海）	ap-shanghai

签名方法 v3

最近更新时间：2025-02-17 15:13:45

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

推荐使用 API Explorer

<> 点击调试

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

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

为什么要进行签名

签名通过以下方式帮助保护请求：

1. 验证请求者的身份

签名确保请求是由持有有效访问密钥的人发送的。请参阅控制台 [云 API 密钥](#) 页面获取密钥相关信息。

2. 保护传输中的数据

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

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

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

申请安全凭证

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

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

申请安全凭证的具体步骤如下：

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

签名版本 v3 签名过程

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

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

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

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

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

```
curl -X POST https://cvm.tencentcloudapi.com \
-H "Authorization: TC3-HMAC-SHA256 Credential=AKID*****/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host;x-tc-action, Signature=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f" \
-H "Content-Type: application/json; charset=utf-8" \
-H "Host: cvm.tencentcloudapi.com" \
-H "X-TC-Action: DescribeInstances" \
-H "X-TC-Timestamp: 1551113065" \
-H "X-TC-Version: 2017-03-12" \
-H "X-TC-Region: ap-guangzhou" \
-d '{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}'
```

下面详细解释签名计算过程。

1. 拼接规范请求串

按如下伪代码格式拼接规范请求串（CanonicalRequest）：

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

字段名称	解释
HTTPRequestMethod	HTTP 请求方法（GET、POST）。此示例取值为 POST。
CanonicalURI	URI 参数，API 3.0 固定为正斜杠（/）。
CanonicalQueryString	发起 HTTP 请求 URL 中的查询字符串，对于 POST 请求，固定为空字符串""，对于 GET 请求，则为 URL 中问号（?）后面的字符串内容，例如：Limit=10&Offset=0。 注意：CanonicalQueryString 需要参考 RFC3986 进行 URLEncode 编码（特殊字符编码后需大写字母），字符集 UTF-8。推荐使用编程语言标准库进行编码。
CanonicalHeaders	参与签名的头部信息，至少包含 host 和 content-type 两个头部，也可加入其他头部参与签名以提高自身请求的唯一性和安全性，此示例额外增加了接口名头部。 拼接规则： 1. 头部 key 和 value 统一转成小写，并去掉首尾空格，按照 key:value\n 格式拼接；

字段名称	解释
	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 <code>content-type:application/json; charset=utf-8\nhost:cvm.tencentcloudapi.com\nx-tc-action:describeinstances\n</code>。 注意：<code>content-type</code> 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 <code>charset</code> 值，如果签名时和发送时不一致，服务器会返回签名校验失败。
SignedHeaders	参与签名的头部信息，说明此次请求有哪些头部参与了签名，和 CanonicalHeaders 包含的头部内容是一一对应的。<code>content-type</code> 和 <code>host</code> 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 <code>content-type;host;x-tc-action</code>
HashedRequestPayload	请求正文（payload，即 body，此示例为 <code>{"Limit": 1, "Filters": [{"Values": [{"\u672a\u547d\u540d"}], "Name": "instance-name"}]}</code>）的哈希值，计算伪代码为 <code>Lowercase(HexEncode(Hash.SHA256(RequestPayload)))</code>，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 <code>35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064</code>。

根据以上规则，示例中得到的规范请求串如下：

```
POST
/

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

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

2. 拼接待签名字符串

按如下格式拼接待签名字符串：

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

字段名称	解释
Algorithm	签名算法，目前固定为 <code>TC3-HMAC-SHA256</code> 。
RequestTimestamp	请求时间戳，即请求头部的公共参数 <code>X-TC-Timestamp</code> 取值，取当前时间 UNIX 时间戳，精确到秒。此示例取值为 <code>1551113065</code> 。
CredentialScope	凭证范围，格式为 <code>Date/service/tc3_request</code> ，包含日期、所请求的服务和终止字符串（ <code>tc3_request</code> ）。 <code>Date</code> 为 UTC 标准时间的日期，取值需要和公共参数 <code>X-TC-Timestamp</code> 换算的

字段名称	解释
	UTC 标准时间日期一致；service 为产品名，必须与调用的产品域名一致。此示例计算结果是 2019-02-25/cvm/tc3_request。
HashedCanonicalRequest	前述步骤拼接所得规范请求串的哈希值，计算伪代码为 Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))。此示例计算结果是 7019a55be8395899b900fb5564e4200d984910f34794a27cb3fb7d10ff6a1e84。

注意：

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

根据以上规则，示例中得到的待签名字符串如下：

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

3. 计算签名

1) 计算派生签名密钥，伪代码如下：

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

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

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

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

2) 计算签名，伪代码如下：

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

此示例计算结果是 10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f。

4. 拼接 Authorization

按如下格式拼接 Authorization:

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

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

根据以上规则，示例中得到的值为：

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

最终完整的调用信息如下：

```
POST https://cvm.tencentcloudapi.com/  
Authorization: TC3-HMAC-SHA256 Credential=AKID*****/2019-02-25/cvm/tc3_request,  
SignedHeaders=content-type;host;x-tc-action, Signature=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b4  
6093f6ab6c4f  
Content-Type: application/json; charset=utf-8  
Host: cvm.tencentcloudapi.com  
X-TC-Action: DescribeInstances  
X-TC-Version: 2017-03-12  
X-TC-Timestamp: 1551113065  
X-TC-Region: ap-guangzhou  
  
{ "Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}
```

⚠ 注意：

请求发送时的 HTTP 头部（Header）和请求体（Payload）必须和签名计算过程中的内容完全一致，否则会返回签名不一致错误。可以通过

打印实际请求内容，网络抓包等方式对比排查。

签名演示

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

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

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

- [Signature Demo](#)

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

Java

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

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

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

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

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

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

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

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

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

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

```
String authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + credentialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
System.out.println(authorization);

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

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

Python

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

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

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

```
params = {"Limit": 1, "Filters": [{"Values": [u"未命名"], "Name": "instance-name"}]}
```

```
# ***** 步骤 1: 拼接规范请求串 *****
```

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

```
# ***** 步骤 2: 拼接待签名字符串 *****
```

```
credential_scope = date + "/" + service + "/" + "tc3_request"
hashed_canonical_request = hashlib.sha256(canonical_request.encode("utf-8")).hexdigest()
string_to_sign = (algorithm + "\n" +
str(timestamp) + "\n" +
credential_scope + "\n" +
hashed_canonical_request)
print(string_to_sign)
```

```
# ***** 步骤 3: 计算签名 *****
```

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

```
# ***** 步骤 4: 拼接 Authorization *****
```

```
authorization = (algorithm + " " +
"Credential=" + secret_id + "/" + credential_scope + ", " +
"SignedHeaders=" + signed_headers + ", " +
"Signature=" + signature)
print(authorization)
```

```
print('curl -X POST ' + endpoint
+ ' -H "Authorization: ' + authorization + '"
+ ' -H "Content-Type: application/json; charset=utf-8"
+ ' -H "Host: ' + host + '"')
```

```
+ ' -H "X-TC-Action: ' + action + '"'  
+ ' -H "X-TC-Timestamp: ' + str(timestamp) + '"'  
+ ' -H "X-TC-Version: ' + version + '"'  
+ ' -H "X-TC-Region: ' + region + '"'  
+ " -d '" + payload + '"")
```

Golang

```
package main  
  
import (  
    "crypto/hmac"  
    "crypto/sha256"  
    "encoding/hex"  
    "fmt"  
    "os"  
    "strings"  
    "time"  
)  
  
func sha256hex(s string) string {  
    b := sha256.Sum256([]byte(s))  
    return hex.EncodeToString(b[:])  
}  
  
func hmacsha256(s, key string) string {  
    hashed := hmac.New(sha256.New, []byte(key))  
    hashed.Write([]byte(s))  
    return string(hashed.Sum(nil))  
}  
  
func main() {  
    // 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****  
    secretId := os.Getenv("TENCENTCLOUD_SECRET_ID")  
    // 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****  
    secretKey := os.Getenv("TENCENTCLOUD_SECRET_KEY")  
    host := "cvm.tencentcloudapi.com"  
    algorithm := "TC3-HMAC-SHA256"  
    service := "cvm"  
    version := "2017-03-12"  
    action := "DescribeInstances"  
    region := "ap-guangzhou"  
    //var timestamp int64 = time.Now().Unix()  
    var timestamp int64 = 1551113065  
  
    // step 1: build canonical request string  
    httpRequestMethod := "POST"  
    canonicalURI := "/"  
    canonicalQueryString := ""
```

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

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

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

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

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

```
fmt.Println(curl)
}
```

PHP

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

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

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

```
.$timestamp."\n"
.$credentialScope."\n"
.$shashedCanonicalRequest;
echo $stringToSign.PHP_EOL;

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

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

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

Ruby

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

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

service = 'cvm'
host = 'cvm.tencentcloudapi.com'
endpoint = 'https://' + host
region = 'ap-guangzhou'
action = 'DescribeInstances'
```

```
version = '2017-03-12'
algorithm = 'TC3-HMAC-SHA256'
# timestamp = Time.now.to_i
timestamp = 1551113065
date = Time.at(timestamp).utc.strftime('%Y-%m-%d')

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

puts canonical_request

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

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

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

```
Signature=#{signature}"
puts authorization

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

DotNet

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

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

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

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

```
DateTime startTime = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc);
long requestTimestamp = (long)Math.Round((date - startTime).TotalMilliseconds, MidpointRounding.AwayFromZero) / 1000;

// ***** 步骤 1: 拼接规范请求串 *****
string algorithm = "TC3-HMAC-SHA256";
string httpRequestMethod = "POST";
string canonicalUri = "/";
string canonicalQueryString = "";
string contentType = "application/json";
string canonicalHeaders = "content-type:" + contentType + "; charset=utf-8\n"
+ "host:" + endpoint + "\n"
+ "x-tc-action:" + action.ToLower() + "\n";
string signedHeaders = "content-type;host;x-tc-action";
string hashedRequestPayload = SHA256Hex(requestPayload);
string canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload;
Console.WriteLine(canonicalRequest);

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

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

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

Dictionary<string, string> headers = new Dictionary<string, string>();
headers.Add("Authorization", authorization);
headers.Add("Host", endpoint);
```

```
headers.Add("Content-Type", contentType + "; charset=utf-8");
headers.Add("X-TC-Timestamp", requestTimestamp.ToString());
headers.Add("X-TC-Version", version);
headers.Add("X-TC-Action", action);
headers.Add("X-TC-Region", region);
return headers;
}

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

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

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

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

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

NodeJS

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

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

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

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

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

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

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

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

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

```
console.log(canonicalRequest)

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

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

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

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

C++

```
#include <algorithm>
#include <cstdlib>
#include <iostream>
#include <iomanip>
#include <sstream>
#include <string>
#include <stdio.h>
#include <time.h>
```

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

using namespace std;

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

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

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

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

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

```
#else
h = HMAC_CTX_new();
#endif

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

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

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

return (ss.str());
}

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

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

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

string service = "cvm";
```

```

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

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

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

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

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

```

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

C

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

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

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

void hmac_sha256(const char* key, int key_len,
const char* input, int input_len,
```

```

unsigned char* output, unsigned int* output_len)
{
    HMAC_CTX *h;
    #if OPENSSL_VERSION_NUMBER < 0x10100000L
    HMAC_CTX hmac;
    HMAC_CTX_init(&hmac);
    h = &hmac;
    #else
    h = HMAC_CTX_new();
    #endif

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

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

}

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

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

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

int main()

```

```
{
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****
const char* SECRET_ID = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
const char* SECRET_KEY = getenv("TENCENTCLOUD_SECRET_KEY");
const char* service = "cvm";
const char* host = "cvm.tencentcloudapi.com";
const char* region = "ap-guangzhou";
const char* action = "DescribeInstances";
const char* version = "2017-03-12";
int64_t timestamp = 1551113065;
char date[20] = {0};
get_utc_date(timestamp, date, sizeof(date));

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

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

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

```
credential_scope, hashed_canonical_request);
printf("%s\n", string_to_sign);

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

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

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

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

其他语言

- Lua: [GitHub](#)
- Swift: [GitHub](#)
- Dart: [GitHub](#)
- Shell(Bash): [GitHub](#)

签名失败

存在以下签名失败的错误码，请根据实际情况处理。

错误码	错误描述
AuthFailure.SignatureExpire	签名过期。Timestamp 与服务器接收到请求的时间相差不得超过五分钟。
AuthFailure.SecretIdNotFound	密钥不存在。请到控制台查看密钥是否被禁用，是否少复制了字符或者多了字符。
AuthFailure.SignatureFailure	签名错误。可能是签名计算错误，或者签名与实际发送的内容不符合，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：2025-02-17 15:13:46

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

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

推荐使用 API Explorer

[<> 点击调试](#)

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

腾讯云 API 会对每个访问请求进行身份验证，即每个请求都需要在公共请求参数中包含签名信息（Signature）以验证请求者身份。

签名信息由安全凭证生成，安全凭证包括 SecretId 和 SecretKey；若用户还没有安全凭证，请前往 [云API密钥页面](#) 申请，否则无法调用云 API 接口。

1. 申请安全凭证

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

安全凭证包括 SecretId 和 SecretKey：

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

申请安全凭证的具体步骤如下：

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

注意：每个账号最多可以拥有两对 SecretId/SecretKey。

2. 生成签名串

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

假设用户的 SecretId 和 SecretKey 分别是：

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

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

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

参数名称	中文	参数值
Action	方法名	DescribeInstances
SecretId	密钥 ID	AKID*****
Timestamp	当前时间戳	1465185768
Nonce	随机正整数	11886

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

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

2.1. 对参数排序

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

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

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

2.2. 拼接请求字符串

此步骤生成请求字符串。

将把上一步排序好的请求参数格式化成“参数名称=参数值”的形式，如对 Action 参数，其参数名称为 "Action"，参数值为 "DescribeInstances"，因此格式化后即为 Action=DescribeInstances。

注意：“参数值”为原始值而非 url 编码后的值。

然后将格式化后的各个参数用"&"拼接在一起，最终生成的请求字符串为：

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

2.3. 拼接签名原文字符串

此步骤生成签名原文字符串。

签名原文字符串由以下几个参数构成：

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

4. 请求字符串: 即上一步生成的请求字符串。

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

示例的拼接结果为:

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

2.4. 生成签名串

此步骤生成签名串。

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

具体代码如下, 以 PHP 语言为例:

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

最终得到的签名串为:

```
7RAM2xfNMO9EiVTNmPg06MRnCvQ=
```

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 7RAM2xfNMO9EiVTNmPg06MRnCvQ=, 最终得到的签名串请求参数 (Signature) 为: 7RAM2xfNMO9EiVTNmPg06MRnCvQ%3D, 它将用于生成最终的请求 URL。

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

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

5. 签名演示

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

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

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

- [Signature Demo](#)

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

最终输出的 url 可能为：`https://cvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKID*****&Signature=7RAM2xfNM09EiVTNmPg06MRnCvQ%3D&Timestamp=1465185768&Version=2017-03-12`。

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

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

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

Java

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

public class TencentCloudAPIDemo {
```

```
private final static String CHARSET = "UTF-8";

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

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

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

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

Python

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

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

import requests

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

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

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

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

Golang

```
package main

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

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

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

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

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

```
}

buf.Truncate(buf.Len() - 1)

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

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

PHP

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

ksort($params);

$signStr = "GETcvm.tencentcloudapi.com/?";
foreach ( $params 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
// $params["Signature"] = $signature;
// $url = "https://cvm.tencentcloudapi.com/?".http_build_query($params);
// echo $url.PHP_EOL;
// $ch = curl_init();
// curl_setopt($ch, CURLOPT_URL, $url);
// $output = curl_exec($ch);
// curl_close($ch);
// echo json_decode($output);
```

Ruby

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

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

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

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

DotNet

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

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

    public static string MakeSignPlainText(SortedDictionary<string, string> requestParams, string requestMethod, string 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
        // 需要设置环境变量 TENCENTCLOUD_SECRET_ID, 值为示例的 AKID*****
        string SECRET_ID = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_ID");
        // 需要设置环境变量 TENCENTCLOUD_SECRET_KEY, 值为示例的 *****
        string SECRET_KEY = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_KEY");

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

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

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

NodeJS

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

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

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

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

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

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

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

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

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

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

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

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

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

main()
```

返回结果

最近更新时间：2025-02-17 15:13:46

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

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

正确返回结果

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

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

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

错误返回结果

若调用失败，其返回值示例如下为：

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

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

公共错误码

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

参数类型

最近更新时间：2025-02-17 15:13:46

目前腾讯云 API 3.0 输入参数和输出参数支持如下几种数据格式：

- String: 字符串。
- Integer: 整型，上限为无符号64位整数。SDK 3.0 不同编程语言支持的类型有所差异，建议以所使用编程语言的最大整型定义，例如 Golang 的 `uint64`。
- Boolean: 布尔型。
- Float: 浮点型。
- Double: 双精度浮点型。
- Date: 字符串，日期格式。例如：2022-01-01。
- Timestamp: 字符串，时间格式。例如：2022-01-01 00:00:00。
- Timestamp ISO8601: ISO 8601 是由国际标准化组织（International Organization for Standardization，ISO）发布的关于日期和时间格式的国际标准，对应国标《GB/T 7408-2005数据元和交换格式信息交换日期和时间表示法》。建议以所使用编程语言的标准库进行格式解析。例如：2022-01-01T00:00:00+08:00。
- Binary: 二进制内容，需要以特定协议请求和解析。

RabbitMQ Serverless 实例管理相关接口

修改RabbitMQ vhost信息

最近更新时间：2025-04-25 02:04:02

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

修改RabbitMQ的vhost

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：ModifyRabbitMQServerlessVirtualHost。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	集群实例Id 示例值：amqp-slrlpeddnr
VirtualHost	是	String	vhost名 示例值：testVhost
Description	否	String	vhost描述信息 示例值：测试 vhost
TraceFlag	否	Boolean	消息轨迹开关,true打开,false关闭 示例值：false

3. 输出参数

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

4. 示例

示例1 成功

输入示例

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

{
  "InstanceId": "amqp-44w9928j",
  "VirtualHost": "testVhost",
  "Description": "test vhost"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "dsfsdfs"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

修改RabbitMQ的用户

最近更新时间：2025-04-25 02:04:02

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

修改RabbitMQ的用户

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：ModifyRabbitMQServerlessUser。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	集群实例Id 示例值：amqp-slrlpeddnr
User	是	String	用户名 示例值：test_user
Password	是	String	密码 示例值：abc123
Description	否	String	描述，不传则不修改 示例值：test user
Tags.N	否	Array of String	用户标签，用于决定改用户访问RabbitMQ Management的权限范围，不传则不修改 示例值：monitoring
MaxConnections	否	Integer	该用户的最大连接数，不传则不修改 示例值：1000
MaxChannels	否	Integer	该用户的最大channel数，不传则不修改 示例值：2000

3. 输出参数

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

4. 示例

示例1 修改RabbitMQ的用户

输入示例

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

{
  "InstanceId": "amqp-44w9928j",
  "User": "test_user",
  "Password": "abcABC123"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "dsfsdfs"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

修改RabbitMQ队列

最近更新时间：2025-04-25 02:04:02

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

修改RabbitMQ队列

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：ModifyRabbitMQServerlessQueue。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	实例Id 示例值：amqp-slrlpeddnr
VirtualHost	是	String	Vhost参数 示例值：testVhost
QueueName	是	String	队列名称 示例值：testQueue
Remark	否	String	新修改的备注 示例值：testRemark

3. 输出参数

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

4. 示例

示例1 修改RabbitMQ队列

输入示例

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

{
  "InstanceId": "amqp-44w9928j",
  "VirtualHost": "testVhost",
  "QueueName": "testQueueName",
  "Remark": "newRemark"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "vcbdasdasdsadsa",
    "QueueName": "cc"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

修改RabbitMQ的权限

最近更新时间：2025-04-25 02:04:02

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

修改RabbitMQ的权限

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：ModifyRabbitMQServerlessPermission。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	集群实例Id 示例值：amqp-slrlpeddnr
User	是	String	用户名，权限关联的用户 示例值：test_user
VirtualHost	是	String	vhost名 示例值：testVhost
ConfigRegexp	是	String	权限类型，declare相关操作，该用户可操作该vhost下的资源名称正则表达式 示例值：.*
WriteRegexp	是	String	权限类型，消息写入相关操作，该用户可操作该vhost下的资源名称正则表达式 示例值：.*
ReadRegexp	是	String	权限类型，消息读取相关操作，该用户可操作该vhost下的资源名称正则表达式 示例值：.*

3. 输出参数

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

4. 示例

示例1 成功

输入示例

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

{
  "InstanceId": "amqp-44w9928j",
  "User": "test_user",
  "VirtualHost": "testVhost",
  "ConfigRegexp": ".*",
  "WriteRegexp": ".*",
  "ReadRegexp": ".*"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "dsfsdfs"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

修改RabbitMQ exchange

最近更新时间：2025-04-25 02:04:02

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

修改RabbitMQ exchange

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：ModifyRabbitMQServerlessExchange。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	实例 id 示例值：amqp-slrlpeddnr
VirtualHost	是	String	vhost 参数 示例值：testVhost
ExchangeName	是	String	exchange 名称 示例值：testExchange
Remark	否	String	备注信息 示例值：测试资源

3. 输出参数

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

4. 示例

示例1 修改RabbitMQ exchange

输入示例

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

{
  "InstanceId": "amqp-44w9928j",
  "VirtualHost": "testVhost",
  "ExchangeName": "testExchange",
  "Remark": "newRemark"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "request1",
    "ExchangeName": "testExchange"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

查询RabbitMQ vhost列表

最近更新时间：2025-04-25 02:04:03

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

查询RabbitMQ vhost列表

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：DescribeRabbitMQServerlessVirtualHost。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	集群实例Id 示例值：amqp-srlpddnr
VirtualHost	否	String	vhost名,不传则查询全部 示例值：testVhost
Offset	否	Integer	分页Offset 示例值：0
Limit	否	Integer	分页Limit 示例值：20
Filters	否	Filter	search-virtual-host：vhost名称模糊查询，之前前缀和后缀匹配 示例值：test
SortElement	否	String	排序依据的字段： MessageHeapCount - 消息堆积数； MessageRateInOut - 生产消费速率之和； MessageRateIn - 生产速率； MessageRateOut - 消费速率； 示例值：MessageHeapCount
SortOrder	否	String	排序顺序，ascend 或 descend 示例值：ascend

3. 输出参数

参数名称	类型	描述
TotalCount	Integer	返回vhost数量 示例值：10
VirtualHostList	Array of RabbitMQVirtualHostInfo	vhost详情列表
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询RabbitMQ vhost列表

输入示例

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

{
  "InstanceId": "amqp-44w9928j"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "dc0de4d7-31c0-47ca-8142-d803d704a44b",
    "TotalCount": 2,
    "VirtualHostList": [
      {
        "InstanceId": "ramqp-cpssyrct",
        "VirtualHost": "testVhost",
        "Description": "default vhost",
        "Tags": [],
        "CreateTime": "2024-10-14 20:06:40:000",
        "ModifyTime": "2024-10-14 20:06:40:000",
        "VirtualHostStatistics": {
          "CurrentQueues": 0,
          "CurrentExchanges": 0,
          "CurrentConnections": 0,
          "CurrentChannels": 0,
          "CurrentUsers": 0
        },
        "Status": "running",
        "MessageHeapCount": 0,

```

```
"MessageRateIn": 0,
"MessageRateOut": 0,
"MirrorQueuePolicyFlag": false
},
{
  "InstanceId": "ramqp-cpssyrct",
  "VirtualHost": "ledou-test",
  "Description": "",
  "Tags": [],
  "CreateTime": "2024-10-16 14:38:58:000",
  "ModifyTime": "2024-10-16 14:38:58:000",
  "VirtualHostStatistics": {
    "CurrentQueues": 0,
    "CurrentExchanges": 0,
    "CurrentConnections": 0,
    "CurrentChannels": 0,
    "CurrentUsers": 0
  },
  "Status": "running",
  "MessageHeapCount": 0,
  "MessageRateIn": 0,
  "MessageRateOut": 0,
  "MirrorQueuePolicyFlag": false
}
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

查询RabbitMQ用户列表

最近更新时间：2025-04-25 02:04:03

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

查询RabbitMQ用户列表

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：DescribeRabbitMQServerlessUser。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	集群实例Id 示例值：amqp-slrlpeddnr
SearchUser	否	String	用户名检索，支持前缀匹配，后缀匹配 示例值：test_user
Offset	否	Integer	分页Offset 示例值：0
Limit	否	Integer	分页Limit 示例值：20
User	否	String	用户名，精确查询 示例值：test_user
Tags.N	否	Array of String	用户标签，根据标签过滤列表 示例值：["tag1","tag2"]

3. 输出参数

参数名称	类型	描述
TotalCount	Integer	返回的User数量 示例值：10
RabbitMQUserList	Array of RabbitMQUser	当前已创建的RabbitMQ用户列表

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

4. 示例

示例1 成功

输入示例

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

{
  "InstanceId": "amqp-44w9928j"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "12bedf1e-5377-46e1-ae05-07225440b16f",
    "TotalCount": 1,
    "RabbitMQUserList": [
      {
        "InstanceId": "ramqp-cpssyrct",
        "User": "testUser",
        "Password": "",
        "Description": "",
        "Tags": [],
        "CreateTime": "2024-10-21 15:52:45:000",
        "ModifyTime": "2024-10-21 15:52:45:000",
        "Type": "default",
        "MaxConnections": null,
        "MaxChannels": null
      }
    ]
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。

错误码	描述
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

查询RabbitMQ队列列表

最近更新时间：2025-04-25 02:04:03

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

查询RabbitMQ队列列表

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：DescribeRabbitMQServerlessQueues。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	实例Id 示例值：amqp-slrlpeddnr
VirtualHost	否	String	Vhost参数 示例值：testVhost
Offset	否	Integer	分页Offset 示例值：0
Limit	否	Integer	分页Limit 示例值：20
SearchWord	否	String	搜索关键词 示例值：testSearchWord
QueueType	否	String	队列类型筛选，不填或 "all"：classic 和 quorum 队列；"classic"：筛选 classic 队列；"quorum"：筛选 quorum 队列 示例值：all
SortElement	否	String	排序依据的字段： ConsumerNumber – 在线消费者数量； MessageHeapCount – 消息堆积数； MessageRateInOut – 生产消费速率之和； MessageRateIn – 生产速率； MessageRateOut – 消费速率； 示例值：MessageHeapCount
SortOrder	否	String	排序顺序，ascend 或 descend 示例值：ascend

3. 输出参数

参数名称	类型	描述
QueueInfoList	Array of RabbitMQQueueListInfo	队列列表信息
TotalCount	Integer	数量 示例值：1
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询RabbitMQ队列列表

输入示例

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

{
  "InstanceId": "amqp-44w9928j",
  "VirtualHost": "testVhost"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "8b5d9dd2-b80f-40da-8c8c-fd180a78be38",
    "TotalCount": 1,
    "QueueInfoList": [
      {
        "QueueName": "ledou-test",
        "Remark": "",
        "ConsumerDetail": {
          "ConsumersNumber": 0
        },
        "QueueType": "normal",
        "MessageHeapCount": 0,
        "MessageRateIn": 0,
        "MessageRateOut": 0,
        "CreateTime": "2024-10-16 14:39:58:000",
        "ModifyTime": "2024-10-16 14:39:58:000",
        "Durable": true,
        "AutoDelete": false,
        "InstanceId": "ramqp-cpssyrct",

```

```
"VirtualHost": "testVhost",
"Node": "",
"Policy": "",
"Arguments": "{\"x-message-ttl\":100000.0}",
"Exclusive": false
}
]
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InternalServerError	内部错误。
InvalidFilter	无效的过滤器
InvalidParameter	参数错误。

错误码	描述
InvalidParameterValue	参数取值错误。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数错误。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

查询RabbitMQ队列详情

最近更新时间：2025-04-25 02:04:03

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

查询RabbitMQ队列详情

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：DescribeRabbitMQServerlessQueueDetail。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	实例Id 示例值：amqp-slrpeddnr
VirtualHost	是	String	Vhost参数 示例值：testVhost
QueueName	是	String	队列名称 示例值：testQueue

3. 输出参数

参数名称	类型	描述
InstanceId	String	实例id 示例值：amqp-slrpeddnr
VirtualHost	String	Vhost参数 示例值：testVhost
QueueName	String	队列名称 示例值：testQueue
QueueType	String	队列类型,取值classic或quorum 示例值：classic
Consumers	Integer	在线消费者数量 示例值：10

参数名称	类型	描述
Durable	Boolean	持久标记 示例值: true
AutoDelete	Boolean	自动清除 示例值: false
Remark	String	备注 示例值: 测试队列
MessageTTL	Integer	MessageTTL 参数,classic类型专用 示例值: 1000
AutoExpire	Integer	AutoExpire参数 示例值: 1000
MaxLength	Integer	MaxLength参数 示例值: 1000
MaxLengthBytes	Integer	MaxLengthBytes参数 示例值: 1000
DeliveryLimit	Integer	DeliveryLimit参数,quorum类型专用 示例值: 1000
OverflowBehaviour	String	OverflowBehaviour参数,取值为drop-head, reject-publish或reject-publish-dlx 示例值: drop-head
DeadLetterExchange	String	DeadLetterExchange参数 示例值: testDeadLetterExchange
DeadLetterRoutingKey	String	DeadLetterRoutingKey参数 示例值: testDeadLetterRoutingKey
SingleActiveConsumer	Boolean	SingleActiveConsumer参数 示例值: true
MaximumPriority	Integer	MaximumPriority参数,classic类型专用 示例值: 1000
LazyMode	Boolean	LazyMode参数,classic类型专用 示例值: true
MasterLocator	String	MasterLocator参数,classic类型专用 示例值: testMasterLocator
MaxInMemoryLength	Integer	MaxInMemoryLength参数,quorum类型专用 示例值: 1000
MaxInMemoryBytes	Integer	MaxInMemoryBytes参数,quorum类型专用 示例值: 1000
CreateTime	Integer	创建时间戳,单位秒 示例值: 1669174535
Node	String	节点 示例值: rabbit@rabbitmq-broker-0.rabbitmq-broker-internal.amqp-xxx.
DeadLetterStrategy	String	仲裁队列死信一致性策略 示例值: at-most-once
QueueLeaderLocator	String	仲裁队列的领导者选举策略

参数名称	类型	描述
		示例值：client-local
QuorumInitialGroupSize	Integer	仲裁队列的初始副本组大小 示例值：3
Exclusive	Boolean	是否为独占队列 示例值：true
Policy	String	生效的策略名 示例值：test-policy
Arguments	String	扩展参数 key-value 示例值：{"x-ordered": false}
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询RabbitMQ队列详情

查询RabbitMQ队列详情

输入示例

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

{
  "InstanceId": "amqp-slmejnrgtz",
  "VirtualHost": "vhost1",
  "QueueName": "queue1"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "e44774dc-597a-45d0-8a41-e5a13bae3878",
    "InstanceId": "amqp-slygmxxjifs",
    "VirtualHost": "test-vhost",
    "QueueName": "queue-test",
    "QueueType": "classic",
    "Consumers": 0,
    "Durable": true,
    "AutoDelete": false,
    "Remark": "",
    "MessageTTL": 100000,
    "AutoExpire": 1,
  }
}
```

```
"MaxLength": 100,
"MaxLengthBytes": 1204,
"DeliveryLimit": 100,
"OverflowBehaviour": "",
"DeadLetterExchange": "",
"DeadLetterRoutingKey": "",
"SingleActiveConsumer": false,
"MaximumPriority": 100,
"LazyMode": false,
"MasterLocator": "",
"MaxInMemoryLength": 1024,
"MaxInMemoryBytes": 1024,
"CreateTime": 1739935376,
"Node": "",
"DeadLetterStrategy": "",
"QueueLeaderLocator": "",
"QuorumInitialGroupSize": 100,
"Exclusive": false,
"Policy": "",
"Arguments": "{\"x-ordered\": false}"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

查询RabbitMQ权限列表

最近更新时间：2025-04-25 02:04:03

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

查询RabbitMQ权限列表

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：DescribeRabbitMQServerlessPermission。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	集群实例id 示例值：amqp-slrlpeddnr
User	否	String	用户名，用于查询过滤，不传则查询全部 示例值：test_user
VirtualHost	否	String	vhost名，用于查询过滤，不传则查询全部 示例值：testVhost
Offset	否	Integer	分页Offset 示例值：0
Limit	否	Integer	分页Limit 示例值：20

3. 输出参数

参数名称	类型	描述
TotalCount	Integer	返回权限数量 示例值：10
RabbitMQPermissionList	Array of RabbitMQPermission	权限详情列表
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询RabbitMQ权限列表

输入示例

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

{
  "InstanceId": "amqp-slfxemaaoa",
  "Offset": 0,
  "Limit": 20,
  "VirtualHost": "",
  "User": ""
}
```

输出示例

```
{
  "Response": {
    "RequestId": "8edba56c-fe02-4e15-8e56-af2a7c4d3760",
    "TotalCount": 1,
    "RabbitMQPermissionList": [
      {
        "InstanceId": "amqp-slfxemaaoa",
        "User": "test_role",
        "VirtualHost": "test_vhost3213",
        "ConfigRegexp": ".*",
        "WriteRegexp": ".*",
        "ReadRegexp": ".*",
        "CreateTime": "2025-01-16 19:52:53",
        "ModifyTime": "2025-01-16 19:52:53"
      }
    ]
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。

错误码	描述
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

查询RabbitMQ exchange 列表

最近更新时间：2025-04-25 02:04:03

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

查询RabbitMQ exchange 列表

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：DescribeRabbitMQServerlessExchanges。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	实例 id 示例值：amqp-slrlpeddnr
VirtualHost	是	String	vhost 参数 示例值：testVhost
Offset	否	Integer	分页 offset 示例值：0
Limit	否	Integer	分页 limit 示例值：20
SearchWord	否	String	搜索关键词, 支持模糊匹配 示例值：testSearchWord
ExchangeTypeFilters.N	否	Array of String	筛选 exchange 类型, 数组中每个元素为选中的过滤类型 示例值：["fanout","direct"]
ExchangeCreatorFilters.N	否	Array of String	筛选 exchange 创建来源, "system":"系统创建", "user":"用户创建" 示例值：["system"]
ExchangeName	否	String	exchange 名称, 用于精确匹配 示例值：exchange1
SortElement	否	String	排序依据的字段： MessageRateInOut – 生产消费速率之和； MessageRateIn – 生产速率； MessageRateOut – 消费速率； 示例值：MessageRateInOut

参数名称	必选	类型	描述
SortOrder	否	String	排序顺序，ascend 或 descend 示例值：descend

3. 输出参数

参数名称	类型	描述
ExchangeInfoList	Array of RabbitMQExchangeListInfo	交换机列表
TotalCount	Integer	交换机总数 示例值：2
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 成功

输入示例

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

{
  "InstanceId": "amqp-44w9928j",
  "VirtualHost": "vhost1"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "request1",
    "ExchangeInfoList": [
      {
        "ExchangeName": "test1",
        "ExchangeType": "fanout",
        "Remark": "test1",
        "VirtualHost": "vhost1",
        "ExchangeCreator": "user",
        "CreateTimeStamp": "2022-12-04 01:08:43",
        "ModTimeStamp": "2022-12-04 01:08:43"
      },
      {
```

```
"ExchangeName": "test2",
"ExchangeType": "direct",
"Remark": "test2",
"VirtualHost": "vhost1",
"ExchangeCreator": "user",
"CreateTimeStamp": "2022-12-04 01:08:43",
"ModTimeStamp": "2022-12-04 01:08:43"
},
"TotalCount": 2
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。

错误码	描述
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

查询RabbitMQ exchange 详情

最近更新时间：2025-04-25 02:04:04

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

查询RabbitMQ exchange 详情

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：DescribeRabbitMQServerlessExchangeDetail。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	实例 id 示例值：amqp-slrlpeddnr
VirtualHost	是	String	vhost 参数 示例值：testVhost
ExchangeName	是	String	exchange 名称 示例值：testExchange

3. 输出参数

参数名称	类型	描述
ExchangeName	String	exchange 名 示例值：testExchangeName
Remark	String	备注说明 示例值：测试交换机
Durable	Boolean	是否为持久化 exchange，当集群重启时，将会清除所有该字段为 "false" 的 exchange 示例值：true
AutoDelete	Boolean	是否自动删除该 exchange，如果为 "true"，当解绑所有当前 exchange 上的路由关系时，该 exchange 将会被自动删除 示例值：false
Internal	Boolean	是否为内部 exchange，如果为 "true"，则无法直接投递消息到该 exchange，需要在路由设置中通过其他 exchange 进行转发 示例值：false

参数名称	类型	描述
AlternateExchange	String	替代 exchange, 如果消息没有匹配当前 exchange 绑定的所有 queue 或 exchange, 就会发送到该替代 exchange 示例值: alternative.exchange1
ExchangeType	String	exchange 类型, 支持 "fanout", "direct", "topic", "headers" 示例值: fanout
VirtualHost	String	VHost参数 示例值: testVhost
ExchangeCreator	String	exchange 创建者, "system":"系统创建", "user":"用户创建" 示例值: user
Arguments	String	扩展参数 key-value 字符串 示例值: map
RequestId	String	唯一请求 ID, 由服务端生成, 每次请求都会返回 (若请求因其他原因未能抵达服务端, 则该次请求不会获得 RequestId)。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查看Exchange详细信息

输入示例

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

```
{
  "InstanceId": "amqp-slfxemaoua",
  "VirtualHost": "test_vhost3213",
  "ExchangeName": "test_exchange"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "e4d30e41-d8c5-4651-81f6-ee286c909030",
    "ExchangeName": "test_exchange",
    "Remark": "",
    "Durable": false,
    "AutoDelete": false,
    "Internal": false,
    "AlternateExchange": "",
    "ExchangeType": "direct",
    "VirtualHost": "test_vhost3213",
    "ExchangeCreator": "user",
```

```
"Arguments": ""
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。

错误码	描述
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

查询RabbitMQ队列消费者列表

最近更新时间：2025-04-25 02:04:04

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

查询RabbitMQ队列消费者列表

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：DescribeRabbitMQServerlessConsumers。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	实例Id 示例值：amqp-slrlpeddnr
VirtualHost	是	String	Vhost参数 示例值：testVhost
QueueName	是	String	队列名 示例值：testQueueName
Limit	否	Integer	分页Limit 示例值：20
Offset	否	Integer	分页Offset 示例值：0
SearchWord	否	String	搜索关键词 示例值：testSearchWord

3. 输出参数

参数名称	类型	描述
ConsumerInfoList	Array of RabbitMQConsumersListInfo	消费者列表信息
TotalCount	Integer	数量 示例值：0

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

4. 示例

示例1 查询RabbitMQ队列消费者列表

输入示例

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

```
{
  "InstanceId": "amqp-jero744g",
  "VirtualHost": "testVhost",
  "QueueName": "testQueue",
  "Limit": 0,
  "Offset": 0,
  "SearchWord": "testSearchWord"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "a8f28d5e-a7e2-4b0b-afa0-2fba09c077a0",
    "ConsumerInfoList": [
      {
        "ClientIp": "119.147.10.198",
        "ConsumerTag": "ctag1.6473a72d0293494197684c6c3261318b"
      }
    ],
    "TotalCount": 1
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。

错误码	描述
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

查询RabbitMQ连接列表

最近更新时间：2025-04-25 02:04:04

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

查询RabbitMQ连接列表

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：DescribeRabbitMQServerlessConnection。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	集群实例Id 示例值：amqp-slrlpeddnr
VirtualHost	是	String	vhost名 示例值：testVhost

3. 输出参数

参数名称	类型	描述
TotalCount	Integer	返回连接数量 示例值：1
Connections	Array of RabbitMQConnection	连接详情列表
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询RabbitMQ连接列表

输入示例

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

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

{
  "InstanceId": "amqp-44w9928j",
  "VirtualHost": "testVhost"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "dsfsdfs",
    "Connections": [],
    "TotalCount": 0
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

查询RabbitMQ路由关系列表

最近更新时间：2025-04-25 02:04:04

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

获取路由关系列表

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：DescribeRabbitMQServerlessBindings。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	实例Id 示例值：amqp-slrlpeddnr
VirtualHost	是	String	Vhost参数 示例值：testVhost
Offset	否	Integer	分页offset 示例值：0
Limit	否	Integer	分页limit 示例值：20
SearchWord	否	String	搜索关键词，根据源exchange名称/目标资源名称/绑定key进行模糊搜索 示例值：testSearchWord
SourceExchange	否	String	根据源Exchange精准搜索过滤 示例值：srcExchange
QueueName	否	String	根据目标QueueName精准搜索过滤，和DestinationExchange过滤不可同时设置 示例值：testQueueName
DestinationExchange	否	String	根据目标Exchange精准搜索过滤，和QueueName过滤不可同时设置 示例值：destExchange

3. 输出参数

参数名称	类型	描述
BindingInfoList	Array of RabbitMQBindingListInfo	路由关系列表
TotalCount	Integer	数量 示例值：10
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询路由绑定信息

输入示例

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

{
  "InstanceId": "amqp-slfxemaoua",
  "VirtualHost": "test_vhost3213",
  "Offset": 0,
  "Limit": 20,
  "SourceExchange": "test_exchange"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "99ff7b6b-bc6e-48e7-8ef6-cffcf74393f6",
    "TotalCount": 1,
    "BindingInfoList": [
      {
        "BindingId": 18921,
        "VirtualHost": "test_vhost3213",
        "Source": "test_exchange",
        "DestinationType": "queue",
        "Destination": "test_queue",
        "RoutingKey": "testRoutingKey",
        "SourceExchangeType": "direct",
        "CreateTime": "2025-01-16 19:52:51",
        "ModifyTime": "2025-01-16 19:52:51"
      }
    ]
  }
}
```

```
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误

错误码	描述
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

删除RabbitMQ的vhost

最近更新时间：2025-04-25 02:04:04

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

删除RabbitMQ的vhost

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：DeleteRabbitMQServerlessVirtualHost。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	集群实例Id 示例值：amqp-slrlpeddnr
VirtualHost	是	String	vhost名 示例值：testVhost

3. 输出参数

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

4. 示例

示例1 成功

输入示例

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

```
{
  "InstanceId": "amqp-44w9928j",
  "VirtualHost": "testVhost"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "dsfsdfs"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。

错误码	描述
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

删除RabbitMQ的用户

最近更新时间：2025-04-25 02:04:04

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

删除RabbitMQ的用户

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：DeleteRabbitMQServerlessUser。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	集群实例Id 示例值：amqp-srlpeddnr
User	是	String	用户名，登录时使用 示例值：test_user

3. 输出参数

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

4. 示例

示例1 成功

输入示例

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

```
{
  "InstanceId": "amqp-xxx",
  "User": "test_user"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "dsfsdfs"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。

错误码	描述
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

删除RabbitMQ队列

最近更新时间：2025-04-25 02:04:04

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

删除RabbitMQ队列

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：DeleteRabbitMQServerlessQueue。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	实例Id 示例值：amqp-slrlpeddnr
VirtualHost	是	String	Vhost参数 示例值：testVhost
QueueName	是	String	队列名称 示例值：testQueue

3. 输出参数

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

4. 示例

示例1 成功

成功

输入示例

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

<公共请求参数>

```
{
  "InstanceId": "amqp-slmejnrgtz",
  "VirtualHost": "vhost1",
  "QueueName": "queue1"
}
```

输出示例

```
{
  "Response": {
    "QueueName": "queue1",
    "RequestId": "7df3c05e-80eb-4d92-b6af-760940b4a2ca"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

删除RabbitMQ的权限

最近更新时间：2025-04-25 02:04:04

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

删除RabbitMQ的权限

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：DeleteRabbitMQServerlessPermission。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	集群实例Id 示例值：amqp-slrlpeddnr
User	是	String	用户名，登录时使用 示例值：test_user
VirtualHost	是	String	vhost名 示例值：testVhost

3. 输出参数

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

4. 示例

示例1 成功

输入示例

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

<公共请求参数>

```
{
  "InstanceId": "amqp-44w9928j",
  "User": "test_user",
  "VirtualHost": "testVhost"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "dsfsdfs"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
DryRunOperation	DryRun 操作，代表请求将会是成功的，只是多传了 DryRun 参数。

错误码	描述
FailedOperation	操作失败。
InternalError	内部错误。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

删除RabbitMQ exchange

最近更新时间：2025-04-25 02:04:05

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

删除RabbitMQ exchange

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：DeleteRabbitMQServerlessExchange。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	实例 id 示例值：amqp-slrlpeddnr
VirtualHost	是	String	vhost 参数 示例值：testVhost
ExchangeName	是	String	exchange 名称 示例值：testExchange

3. 输出参数

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

4. 示例

示例1 成功

输入示例

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

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

{
  "InstanceId": "amqp-slrlpeddnr",
  "VirtualHost": "testVhost",
  "ExchangeName": "testExchange"
}
```

输出示例

```
{
  "Response": {
    "ExchangeName": "testExchange",
    "RequestId": "a8f28d5e-a7e2-4b0b-afa0-2fba09c077a0"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

解绑RabbitMQ路由关系

最近更新时间：2025-04-25 02:04:05

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

解绑RabbitMQ路由关系

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：DeleteRabbitMQServerlessBinding。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	实例Id 示例值：amqp-slrlpeddnr
VirtualHost	是	String	Vhost参数 示例值：testVhost
BindingId	是	Integer	路由关系Id 示例值：127469

3. 输出参数

参数名称	类型	描述
InstanceId	String	队列名称 示例值：amqp-slrlpeddnr
VirtualHost	String	vhost参数 示例值：testVhost
BindingId	Integer	路由关系Id 示例值：127469
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 解绑RabbitMQ路由关系

输入示例

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

{
  "InstanceId": "amqp-44w9928j",
  "VirtualHost": "testVhost",
  "BindingId": 127441
}
```

输出示例

```
{
  "Response": {
    "RequestId": "cxvxcvcxfdsfds",
    "InstanceId": "amqp-44w9928j",
    "VirtualHost": "testVhost",
    "BindingId": 127441
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

创建RabbitMQ的vhost

最近更新时间：2025-04-25 02:04:05

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

创建RabbitMQ的vhost

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：CreateRabbitMQServerlessVirtualHost。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	集群实例Id 示例值：amqp-slrlpeddnr
VirtualHost	是	String	vhost名称 示例值：testVhost
Description	否	String	描述信息 示例值：测试vhost
TraceFlag	否	Boolean	消息轨迹开关,true打开,false关闭,默认关闭 示例值：true
MirrorQueuePolicyFlag	否	Boolean	是否创建镜像队列策略，默认值 true 示例值：true

3. 输出参数

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

4. 示例

示例1 成功

输入示例

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

{
  "InstanceId": "amqp-44w9928j",
  "VirtualHost": "testVhost",
  "Description": "hello"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "dsfsdfs",
    "VirtualHost": "testVhost"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

创建RabbitMQ的用户

最近更新时间：2025-04-25 02:04:05

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

创建RabbitMQ的用户

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：CreateRabbitMQServerlessUser。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	集群实例Id 示例值：amqp-slrlpeddnr
User	是	String	用户名，登录时使用 示例值：test_user
Password	是	String	密码，登录时使用 示例值：abc123
Description	否	String	描述 示例值：测试用户
Tags.N	否	Array of String	serverless 实例该字段无效 示例值：monitoring
MaxConnections	否	Integer	该用户的最大连接数，不填写则不限制 示例值：1000
MaxChannels	否	Integer	该用户的最大channel数，不填写则不限制 示例值：2000

3. 输出参数

参数名称	类型	描述
User	String	用户名，登录时使用 示例值：test_user

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

4. 示例

示例1 成功

输入示例

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

{
  "InstanceId": "amqp-44w9928j",
  "User": "test_user",
  "Password": "abc123"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "dsfsdfs",
    "User": "test_user"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Java: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for PHP: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Go: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Node.js: [GitHub](#), [Gitee](#)

- Tencent Cloud SDK 3.0 for .NET: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for C++: [GitHub](#), [Gitee](#)
- Tencent Cloud SDK 3.0 for Ruby: [GitHub](#), [Gitee](#)

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

创建RabbitMQ队列

最近更新时间：2025-04-25 02:04:05

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

创建RabbitMQ队列

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：CreateRabbitMQServerlessQueue。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	实例Id 示例值：amqp-slrlpeddnr
VirtualHost	是	String	VHost参数 示例值：testVhost
QueueName	是	String	队列名称 示例值：testQueue
QueueType	否	String	只支持 classic 示例值：classic
Durable	否	Boolean	持久标记,classic类型必传,quorum类型无需传入固定为true 示例值：true
AutoDelete	否	Boolean	自动清除,classic类型必传,quorum类型无需传入固定为false 示例值：false
Remark	否	String	备注 示例值：测试队列
MessageTTL	否	Integer	MessageTTL参数,classic类型专用 示例值：1000
AutoExpire	否	Integer	AutoExpire参数，单位为 ms，队列在指定时间内没有被使用，将会被删除 示例值：1000
MaxLength	否	Integer	MaxLength参数。队列可以容纳的最大条数。若超出上限，将根据 overview behavior 处理 示例值：1000

参数名称	必选	类型	描述
MaxLengthBytes	否	Integer	MaxLengthBytes参数。若超出上限，将根据 overview behavior 处理。 示例值：1000
DeliveryLimit	否	Integer	DeliveryLimit参数,quorum类型专用 示例值：1000
OverflowBehaviour	否	String	OverflowBehaviour参数,取值为drop-head, reject-publish或reject-publish-dlx 示例值：drop-head
DeadLetterExchange	否	String	DeadLetterExchange参数。可将过期或被拒绝的消息投往指定的死信 exchange。 示例值：testDeadLetterExchange
DeadLetterRoutingKey	否	String	DeadLetterRoutingKey参数。只能包含字母、数字、"."、"-", "@", "_" 示例值：testDeadLetterRoutingKey
SingleActiveConsumer	否	Boolean	SingleActiveConsumer参数。若开启，需确保每次有且只有一个消费者从队列中消费 示例值：true
MaximumPriority	否	Integer	MaximumPriority参数,classic类型专用 示例值：1000
LazyMode	否	Boolean	LazyMode参数,classic类型专用 示例值：true
MasterLocator	否	String	MasterLocator参数,classic类型专用,取值为min-masters,client-local或random 示例值：testMasterLocator
MaxInMemoryLength	否	Integer	MaxInMemoryLength参数，quorum类型专用。quorum 队列的内存中最大消息数量 示例值：1000
MaxInMemoryBytes	否	Integer	MaxInMemoryBytes参数，quorum类型专用。quorum 队列的内存中最大数总消息大小 示例值：1000
Node	否	String	Node参数，非必填，指定创建 queue 所在节点 示例值：rabbit@amqp-xxxxxxx-1
DeadLetterStrategy	否	String	仲裁队列死信一致性策略，at-most-once、at-least-once，默认是at-most-once 示例值：at-most-once
QueueLeaderLocator	否	String	仲裁队列的领导者选举策略，client-local、balanced，默认是client-local 示例值：client-local
QuorumInitialGroupSize	否	Integer	仲裁队列的初始副本组大小，默认3 示例值：3

3. 输出参数

参数名称	类型	描述
QueueName	String	队列名称 示例值：queue1
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 创建RabbitMQ队列

创建RabbitMQ队列

输入示例

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

{
  "InstanceId": "amqp-slmejnrgtz",
  "VirtualHost": "vhost1",
  "QueueName": "queue1",
  "QueueType": "vhost1",
  "Remark": "测试队列"
}
```

输出示例

```
{
  "Response": {
    "QueueName": "queue1",
    "RequestId": "941a7f2b-26f7-413f-b91f-341678cd00f6"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

- Tencent Cloud SDK 3.0 for Ruby: [GitHub](#), [Gitee](#)

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

创建RabbitMQ exchange

最近更新时间：2025-04-25 02:04:05

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

创建RabbitMQ exchange

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：CreateRabbitMQServerlessExchange。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	实例Id 示例值：amqp-slrlpeddnr
VirtualHost	是	String	VHost参数 示例值：testVhost
ExchangeName	是	String	exchange 名称 示例值：testExchange
ExchangeType	是	String	exchange 类型，支持 "fanout","direct","topic","headers" 示例值：fanout
Remark	否	String	exchange 备注 示例值：测试交换机
Durable	否	Boolean	是否为持久化 exchange，当集群重启时,将会清除所有该字段为"false"的 exchange 示例值：true
AutoDelete	否	Boolean	是否自动删除该 exchange，如果为 "true"，当解绑所有当前 exchange 上的路由关系时，该 exchange 将会被自动删除 示例值：false
Internal	否	Boolean	是否为内部 exchange，如果为 "true"，则无法直接投递消息到该 exchange，需要在路由设置中通过其他 exchange 进行转发 示例值：false
AlternateExchange	否	String	替代 exchange，如果消息无法发送到当前 exchange，就会发送到该替代 exchange 示例值：alternative.exchange1

3. 输出参数

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

4. 示例

示例1 成功

成功

输入示例

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

{
  "InstanceId": "amqp-slmejnrqtz",
  "VirtualHost": "vhost1",
  "ExchangeName": "exchange2",
  "ExchangeType": "fanout",
  "Remark": "测试交换机",
  "Durable": true
}
```

输出示例

```
{
  "Response": {
    "ExchangeName": "exchange2",
    "RequestId": "01f525e3-fc29-498e-a801-741cee1aed9e"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

创建RabbitMQ路由关系

最近更新时间：2025-04-25 02:04:05

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

创建RabbitMQ路由关系

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：CreateRabbitMQServerlessBinding。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	实例Id 示例值：amqp-slrlpeddnr
VirtualHost	是	String	Vhost参数 示例值：testVhost
Source	是	String	源exchange 示例值：source-exchange
DestinationType	是	String	目标类型,取值queue或exchange 示例值：queue
Destination	是	String	目标队列或者交换机 示例值：des-queue
RoutingKey	否	String	绑定key 示例值：some-routing-key

3. 输出参数

参数名称	类型	描述
InstanceId	String	队列名称 示例值：amqp-slrlpeddnr
VirtualHost	String	vhost参数 示例值：testVhost

参数名称	类型	描述
BindingId	Integer	路由关系Id 示例值：1000012
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 添加绑定关系

添加绑定关系

输入示例

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

{
  "InstanceId": "amqp-slnatbyves",
  "VirtualHost": "vhost",
  "Source": "fanout-exchange-1061997075421934168",
  "DestinationType": "queue",
  "Destination": "queue-623992266636621729"
}
```

输出示例

```
{
  "Response": {
    "BindingId": 0,
    "InstanceId": "amqp-slnatbyves",
    "RequestId": "1dd2c068-0d69-4ec4-a9e5-adce8cffa43b",
    "VirtualHost": "vhost"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

RabbitMQ Serverless PAAS 能力相关接口

修改 rabbitmq serverless 实例信息

最近更新时间：2025-04-25 02:04:05

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

修改集群信息

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

</> 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数 ，本接口取值：ModifyRabbitMQServerlessInstance。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	实例id 示例值：amqp-slrlpeddnr
ClusterName	否	String	集群名 示例值：test-cluster
Remark	否	String	备注信息 示例值：测试集群
TraceFlag	否	Boolean	是否开启trace 示例值：false

3. 输出参数

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

4. 示例

示例1 修改集群信息

修改集群信息

输入示例

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

{
  "InstanceId": "amqp-slmejnrgtz",
  "ClusterName": "dev-env",
  "Remark": "测试修改"
}
```

输出示例

```
{
  "Response": {
    "InstanceId": "amqp-slmejnrgtz",
    "RequestId": "ec77f16a-1349-4014-9eda-1cdc0b35f341"
  }
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

获取 RabbitMQ Serverless 集群列表

最近更新时间：2025-05-29 02:07:03

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

获取实例列表

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：ListRabbitMQServerlessInstances。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
Filters.N	否	Array of Filter	过滤条件
Limit	否	Integer	翻页大小 示例值：10
Offset	否	Integer	翻页的起始索引值 示例值：0

3. 输出参数

参数名称	类型	描述
Instances	Array of RabbitMQServerlessInstance	实例列表
TotalCount	Integer	总数 示例值：10
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 获取实例列表

输入示例

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

<公共请求参数>

```
{
  "Filters": [],
  "Limit": 2,
  "Offset": 1
}
```

输出示例

```
{
  "Response": {
    "Instances": [
      {
        "AutoRenewFlag": 1,
        "ClusterStatus": 1,
        "CreateTime": 1737028115000,
        "ExceptionInformation": "",
        "ExpireTime": 1739706515000,
        "InstanceId": "amqp-slfxemaaoa",
        "InstanceName": "test_yunchao_8816",
        "InstanceType": 1,
        "InstanceVersion": "0.1.0",
        "MaxBandWidth": 0,
        "MaxStorage": 0,
        "MaxTps": 2000,
        "NodeCount": 0,
        "PayMode": 0,
        "PublicAccessEndpoint": "",
        "Remark": "",
        "SpecName": "基础版",
        "Status": 1,
        "Vpcs": []
      },
      {
        "AutoRenewFlag": 1,
        "ClusterStatus": 1,
        "CreateTime": 1736912606000,
        "ExceptionInformation": "",
        "ExpireTime": 1739591006000,
        "InstanceId": "amqp-slirnlmwlm",
        "InstanceName": "test_yunchao_001",
        "InstanceType": 1,
        "InstanceVersion": "0.1.0",
        "MaxBandWidth": 0,
```

```
"MaxStorage": 0,
"MaxTps": 2000,
"NodeCount": 0,
"PayMode": 0,
"PublicAccessEndpoint": "",
"Remark": "",
"SpecName": "基础版",
"Status": 1,
"Vpcs": [
{
"SubnetId": "subnet-1j1p6omq",
"VpcDataStreamEndpointStatus": "ON",
"VpcEndpoint": "amqp://10.0.0.2:5672",
"VpcId": "vpc-f7zemc2v"
}
],
"RequestId": "cc77d367-76b7-4aef-b5d6-e471ed254758",
"TotalCount": 4
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
MissingParameter	缺少参数错误。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。

获取 rabbitmq serverless 实例总体信息

最近更新时间：2025-04-25 02:04:06

1. 接口描述

接口请求域名：trabbit.tencentcloudapi.com。

获取单个RabbitMQ专享实例信息

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

</> 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数 ，本接口取值：DescribeRabbitMQServerlessInstance。
Version	是	String	公共参数 ，本接口取值：2023-04-18。
Region	是	String	公共参数 ，详见产品支持的 地域列表 。
InstanceId	是	String	集群ID 示例值：amqp-slrlpeddnr

3. 输出参数

参数名称	类型	描述
ClusterInfo	RabbitMQClusterInfo	集群信息
ClusterSpecInfo	RabbitMQClusterSpecInfo	集群规格信息
VirtualHostQuota	VirtualHostQuota	vhost配额信息
ExchangeQuota	ExchangeQuota	exchange配额信息
QueueQuota	QueueQuota	queue配额信息
ClusterNetInfo	RabbitMQServerlessAccessInfo	网络信息
ClusterWhiteListInfo	RabbitMQServerlessWhiteListInfo	公网白名单信息
UserQuota	UserQuota	user配额信息
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 实例详情示例

输入示例

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

{
  "InstanceId": "amqp-slnatby"
}
```

输出示例

```
{
  "Response": {
    "RequestId": "bae00352-45fb-46ea-8960-2c8ce102db5f",
    "ClusterInfo": {
      "InstanceType": 1,
      "ClusterName": "devCluster",
      "ClusterId": "amqp-slnatby",
      "Region": "ap-guangzhou",
      "CreateTime": 1737775135000,
      "Remark": "",
      "Vpcs": [
        {
          "VpcId": "vpc-test",
          "SubnetId": "subnet-test",
          "VpcEndpoint": "amqp://10.0.0.1:5672",
          "VpcDataStreamEndpointStatus": "ON"
        }
      ],
      "ZoneIds": null,
      "VirtualHostNumber": 2,
      "QueueNumber": 11,
      "ChannelNumber": 0,
      "ConnectionNumber": 0,
      "ConsumerNumber": 0,
      "ExchangeNumber": 13,
      "MessagePublishRate": 0,
      "MessageConsumeRate": 0,
      "MessageStackNumber": 19,
      "ExpireTime": 1740453535000,
      "ExceptionInformation": null,
      "ClusterStatus": 1,
      "AutoRenewFlag": 1,
      "ClusterVersion": "0.1.0",
      "PayMode": 0,
```

```
"MessageRetainTime": 72,
"MirrorQueuePolicyFlag": null
},
"ClusterNetInfo": {
  "PublicDataStreamStatus": "OFF",
  "PublicAccessEndpoint": null
},
"ClusterWhiteListInfo": {
  "PublicDataStreamWhiteList": null,
  "PublicDataStreamWhiteListStatus": "OFF"
},
"ClusterSpecInfo": {
  "SpecName": "BASIC",
  "MaxTps": 2000,
  "MaxQueueNum": 100,
  "MaxVhostNum": 250,
  "MaxExchangeNum": 100,
  "MaxConnNum": 2000,
  "MaxUserNum": 20,
  "MaxBandWidth": 0,
  "PublicNetworkTps": 3
},
"VirtualHostQuota": {
  "MaxVirtualHost": 250,
  "UsedVirtualHost": 2
},
"ExchangeQuota": {
  "MaxExchange": 100,
  "UsedExchange": 13
},
"QueueQuota": {
  "MaxQueue": 100,
  "UsedQueue": 11
},
"UserQuota": {
  "MaxUser": 20,
  "UsedUser": 1
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

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

命令行工具

- [Tencent Cloud CLI 3.0](#)

6. 错误码

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

错误码	描述
AuthFailure	CAM签名/鉴权错误。
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。
InvalidFilter	无效的过滤器
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。
OperationDenied	操作被拒绝。
RegionError	地域错误
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。
ResourcesSoldOut	资源售罄。
UnauthorizedOperation	未授权操作。

错误码	描述
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。

数据结构

最近更新时间：2025-05-29 02:07:03

ExchangeQuota

exchange使用配额信息

被如下接口引用：DescribeRabbitMQServerlessInstance。

名称	类型	描述
MaxExchange	Integer	可创建最大exchange数 示例值：10
UsedExchange	Integer	已创建exchange数 示例值：5

Filter

过滤参数

被如下接口引用：DescribeRabbitMQServerlessVirtualHost, ListRabbitMQServerlessInstances。

名称	类型	必选	描述
Name	String	否	过滤参数的名字 示例值：instanceName
Values	Array of String	否	数值 示例值：amqp-slpvmqwcna

QueueQuota

queue使用配额信息

被如下接口引用：DescribeRabbitMQServerlessInstance。

名称	类型	描述
MaxQueue	Integer	可创建最大Queue数 示例值：10
UsedQueue	Integer	已创建Queue数 示例值：5

RabbitMQBindingListInfo

Rabbitmq路由关系列表成员

被如下接口引用：DescribeRabbitMQServerlessBindings。

名称	类型	描述
BindingId	Integer	路由关系id 示例值：127469
VirtualHost	String	Vhost参数 示例值：testVhost

名称	类型	描述
Source	String	源exchange名称 示例值：testSource
DestinationType	String	目标类型,queue或exchange 示例值：queue
Destination	String	目标资源名称 示例值：testDestination
RoutingKey	String	绑定key 示例值：testRoutingKey
SourceExchangeType	String	源exchange类型 示例值：testSourceExchangeType
CreateTime	String	创建时间 示例值：2023-11-20 11:15:47.612
ModifyTime	String	修改时间 示例值：2023-11-20 11:15:47.612

RabbitMQClusterInfo

RabbiteMQ集群基本信息

被如下接口引用：DescribeRabbitMQServerlessInstance。

名称	类型	描述
ClusterId	String	集群ID 示例值：amqp-slrlpeddnr
ClusterName	String	集群名称 示例值：测试集群
Region	String	地域信息 示例值：ap-guangzhou
CreateTime	Integer	创建时间，毫秒为单位 示例值：1728907454000
Remark	String	集群说明信息 示例值：dev开发环境
Vpcs	Array of VpcEndpointInfo	VPC及网络信息
Zonelds	Array of Integer	可用区信息 示例值：[190001]
VirtualHostNumber	Integer	虚拟主机数量 示例值：10
QueueNumber	Integer	队列数量 示例值：10
MessagePublishRate	Float	每秒生产消息数 单位：条/秒 示例值：2.3

名称	类型	描述
MessageStackNumber	Integer	堆积消息数 单位：条 示例值：5
ExpireTime	Integer	过期时间 示例值：1666257531143
ChannelNumber	Integer	Channel数量 示例值：10
ConnectionNumber	Integer	Connection数量 示例值：10
ConsumerNumber	Integer	Consumer数量 示例值：10
ExchangeNumber	Integer	Exchang数量 示例值：10
ExceptionInformation	String	集群异常信息 示例值：异常信息
ClusterStatus	Integer	实例状态，0表示创建中，1表示正常，2表示隔离中，3表示已销毁，4 – 异常, 5 – 发货失败 示例值：0
AutoRenewFlag	Integer	自动续费标记，0表示默认状态(用户未设置，即初始状态即手动续费)， 1表示自动续费，2表示明确不自动续费(用户设置) 示例值：1
MirrorQueuePolicyFlag	Integer	是否开启镜像队列策略。1表示开启，0表示没开启。 示例值：1
MessageConsumeRate	Float	每秒消费消息数 单位：条/秒 示例值：1.0
ClusterVersion	String	集群版本信息 示例值：3.8.30
PayMode	Integer	计费模式，0-后付费，1-预付费 示例值：0
InstanceType	Integer	集群类型 示例值：1
MessageRetainTime	Integer	消息保留时间，单位小时 示例值：72

RabbitMQClusterSpecInfo

RabbitMQ集群规格信息

被如下接口引用：DescribeRabbitMQServerlessInstance。

名称	类型	描述
SpecName	String	集群规格名称 示例值：BASIC
MaxTps	Integer	峰值tps 示例值：5000

名称	类型	描述
MaxQueueNum	Integer	最大队列数 示例值：500
MaxExchangeNum	Integer	最大交换机数 示例值：500
MaxVhostNum	Integer	最大vhost数 示例值：100
MaxConnNum	Integer	最大连接数 示例值：10000
MaxUserNum	Integer	最大用户数 示例值：20
MaxBandWidth	Integer	峰值带宽，已废弃 示例值：3
PublicNetworkTps	Integer	公网带宽，已废弃 示例值：3
Features	String	实例对应的功能列表，true表示支持，false 表示不支持 示例值：{"billing":true,"monitor":true}

RabbitMQConnection

RabbitMQ连接详情

被如下接口引用：DescribeRabbitMQServerlessConnection。

名称	类型	描述
ConnectionName	String	连接名称 示例值：1.1.1.1:2340 -> 10.1.1.1:5672
PeerHost	String	客户端ip 示例值：1.1.1.1
State	String	连接状态，包括 starting、tuning、opening、running、flow、blocking、blocked、closing 和 closed 示例值：running
User	String	连接使用用户 示例值：admin
SSL	Boolean	是否开启ssl 示例值：false
Protocol	String	连接协议 示例值：AMQP 0-9-1
Channels	Integer	连接下的channel数 示例值：1

RabbitMQConsumersListInfo

队列消费者列表信息

被如下接口引用：DescribeRabbitMQServerlessConsumers。

名称	类型	描述
ClientIp	String	客户端Ip 示例值：1.1.1.1
ConsumerTag	String	消费者Tag 示例值：testConsumerTag

RabbitMQExchangeListInfo

RabbitMQ exchange列表成员信息

被如下接口引用：DescribeRabbitMQServerlessExchanges。

名称	类型	描述
ExchangeName	String	exchange 名 示例值：testExchangeName
Remark	String	备注说明 示例值：testRemark
ExchangeType	String	exchange 类型, 支持 "fanout", "direct", "topic", "headers" 示例值：fanout
VirtualHost	String	VHost参数 示例值：testVhost
ExchangeCreator	String	exchange 创建者, "system":"系统创建", "user":"用户创建" 示例值：user
CreateTimeStamp	String	exchange 创建时间 示例值：2016-11-09 11:44:44
ModTimeStamp	String	exchange 修改时间 示例值：2016-11-09 11:44:44
MessageRateIn	Float	输入消息速率 示例值：0.0
MessageRateOut	Float	输出消息速率 示例值：0.0
Durable	Boolean	是否为持久化交换机, true 为持久化, false 为非持久化 示例值：true
AutoDelete	Boolean	是否为自动删除交换机, true 为自动删除, false 为非自动删除 示例值：true
Internal	Boolean	是否为内部交换机, true 为内部交换机 示例值：true
InstanceId	String	交换机所属实例 ID 示例值：amqp-xxx
Policy	String	生效的策略名称 示例值：test-policy
Arguments	String	扩展参数 key-value 对象 示例值：{"x-ordered": false}

名称	类型	描述
MessagesDelayed	Integer	未调度的延时消息数量 示例值：0

RabbitMQPermission

RabbitMQ权限详情

被如下接口引用：DescribeRabbitMQServerlessPermission。

名称	类型	描述
InstanceId	String	集群实例Id 示例值：amqp-xxx
User	String	用户名，权限关联的用户 示例值：test_user
VirtualHost	String	vhost名 示例值：testVhost
ConfigRegexp	String	权限类型，declare相关操作，该用户可操作该vhost下的资源名称正则表达式 示例值：.*
WriteRegexp	String	权限类型，消息写入相关操作，该用户可操作该vhost下的资源名称正则表达式 示例值：.*
ReadRegexp	String	权限类型，消息读取相关操作，该用户可操作该vhost下的资源名称正则表达式 示例值：.*
CreateTime	String	创建时间 示例值：2023-08-29 14:06:38.067
ModifyTime	String	修改时间 示例值：2023-08-29 14:06:38.067

RabbitMQQueueListConsumerDetailInfo

RabbitMQ队列列表消费者信息

被如下接口引用：DescribeRabbitMQServerlessQueues。

名称	类型	描述
ConsumersNumber	Integer	消费者数量 示例值：1

RabbitMQQueueListInfo

RabbitMQ队列列表成员信息

被如下接口引用：DescribeRabbitMQServerlessQueues。

名称	类型	描述
QueueName	String	队列名 示例值：testQueueName

名称	类型	描述
Remark	String	备注说明 示例值：testRemark
ConsumerDetail	RabbitMQQueueListConsumerDetailInfo	消费者信息
QueueType	String	队列类型，取值 "classic"，"quorum" 示例值：classic
MessageHeapCount	Integer	消息堆积数 示例值：20
MessageRateIn	Float	消息生产速率，每秒 示例值：0.0
MessageRateOut	Float	消息消费速率，每秒 示例值：0.0
CreateTime	String	创建时间 示例值：2024-01-03 11:58:39.447
ModifyTime	String	修改时间 示例值：2024-01-03 11:58:39.447
Durable	Boolean	队列是否持久化，true 为持久化，false 为非持久化 示例值：true
AutoDelete	Boolean	队列是否为自动删除队列，true 为自动删除，false 为非自动删除 示例值：true
Instanceld	String	队列所属实例 ID 示例值：amqp-xxx
VirtualHost	String	队列所属虚拟主机名称 示例值：testVhost
Node	String	队列所在主节点名称 示例值：rabbit@amqp-xxx.rabbitmq-broker-0
Policy	String	生效的策略名称 示例值：test-policy
Arguments	String	扩展参数 key-value 对象 示例值：{"x-ordered": false}
Exclusive	Boolean	是否独占队列 示例值：false

RabbitMQServerlessAccessInfo

公网访问信息

被如下接口引用：DescribeRabbitMQServerlessInstance。

名称	类型	描述
PublicAccessEndpoint	String	公网域名 示例值：http://127.4.5.1

名称	类型	描述
PublicDataStreamStatus	String	公网状态 示例值：on

RabbitMQServerlessEndpoint

接入点

被如下接口引用：ListRabbitMQServerlessInstances。

名称	类型	描述
VpcId	String	vpc id 示例值：vpc-5ghsr4p9
SubnetId	String	subnet id 示例值：subnet-67y9wi
VpcEndpoint	String	接入地址 示例值：http://145.10.3.4
VpcDataStreamEndpointStatus	String	接入地址状态 示例值：on
PublicNetwork	Boolean	是否是公网 示例值：true
AccessStrategy	String	访问策略 示例值：127.5.4.1
Bandwidth	Integer	带宽 示例值：20

RabbitMQServerlessInstance

rabbitmq serverless 实例

被如下接口引用：ListRabbitMQServerlessInstances。

名称	类型	描述
InstanceId	String	实例Id 示例值：amqp-xxxxxxx
InstanceName	String	实例名称 示例值：测试实例
InstanceVersion	String	实例版本号 示例值：1.0.0
Status	Integer	实例状态，0表示创建中，1表示正常，2表示隔离中，3表示已销毁，4 - 异常，5 - 发货失败 示例值：集群状态
MaxTps	Integer	TPS 示例值：500
MaxBandWidth	Integer	带宽 示例值：10

名称	类型	描述
ExpireTime	Integer	集群过期时间 示例值：1738330089000
AutoRenewFlag	Integer	自动续费标记，0表示默认状态(用户未设置，即初始状态即手动续费)，1表示自动续费，2表示明确不自动续费(用户设置) 示例值：1
PayMode	Integer	0-后付费，1-预付费 示例值：0
Remark	String	备注 示例值：测试集群
SpecName	String	集群规格 示例值：专业版
ExceptionInformation	String	异常信息 示例值：获取列表失败
PublicAccessEndpoint	String	公网接入点 示例值：http://127.1.1.1
Vpcs	Array of RabbitMQServerlessEndpoint	私有网络接入点
ClusterStatus	Integer	实例状态，0表示创建中，1表示正常，2表示隔离中，3表示已销毁，4 - 异常，5 - 发货失败 示例值：1
InstanceType	Integer	集群类型：1 示例值：1
CreateTime	Integer	过期时间 示例值：1738330089000
NodeCount	Integer	为了兼容托管版，固定值 0 示例值：0
MaxStorage	Integer	为了兼容托管版，固定值 0 示例值：0
IsolatedTime	Integer	隔离时间 示例值：1738330089000
ServerlessExt	String	Serverless 扩展字段 示例值：{"tps":"2000"}

RabbitMQServerlessWhiteListInfo

公网白名单信息

被如下接口引用：DescribeRabbitMQServerlessInstance。

名称	类型	描述
PublicDataStreamWhiteList	String	公网数据流白名单 示例值：1.2.3.4,5.6.7.8

名称	类型	描述
PublicDataStreamWhiteListStatus	String	公网数据流白名单状态 示例值：OFF/ON/MODIFYING

RabbitMQUser

RabbitMQ用户实体详情

被如下接口引用：DescribeRabbitMQServerlessUser。

名称	类型	描述
InstanceId	String	集群实例Id 示例值：amqp-xxx
User	String	用户名，登录时使用 示例值：test_user
Password	String	密码，登录时使用 示例值：abc123
Description	String	用户描述 示例值：测试账号
Tags	Array of String	用户标签，用于决定改用户访问RabbitMQ Management的权限范围 示例值：monitoring
CreateTime	String	用户创建时间 示例值：2022-11-11 11:11:11
ModifyTime	String	用户最后修改时间 示例值：2022-11-11 11:11:11
Type	String	用户类型，System：系统创建，User：用户创建 示例值：System
MaxConnections	Integer	该用户所能允许的最大连接数 示例值：10
MaxChannels	Integer	该用户所能允许的最大通道数 示例值：1000

RabbitMQVirtualHostInfo

RabbitMQ的vhost详情

被如下接口引用：DescribeRabbitMQServerlessVirtualHost。

名称	类型	描述
InstanceId	String	集群实例Id 示例值：amqp-xxx
VirtualHost	String	vhost名 示例值：testVhost
Description	String	vhost描述信息 示例值：hello

名称	类型	描述
Tags	Array of String	vhost标签 示例值：shop,account
CreateTime	String	创建时间 示例值：2022-11-11 11:11:11
ModifyTime	String	修改时间 示例值：2022-11-11 11:11:11
VirtualHostStatistics	RabbitMQVirtualHostStatistics	vhost概览统计信息
Status	String	vhost状态，与原生控制台对应，有running、partial、stopped、unknown 示例值：running
MessageHeapCount	Integer	消息堆积数 示例值：0
MessageRateIn	Float	输入消息速率 示例值：0
MessageRateOut	Float	输出消息速率 示例值：0
MirrorQueuePolicyFlag	Boolean	是否存在镜像队列策略，true 为存在，false 为不存在 示例值：true

RabbitMQVirtualHostStatistics

vhost概览统计信息

被如下接口引用：DescribeRabbitMQServerlessVirtualHost。

名称	类型	描述
CurrentQueues	Integer	当前vhost的queue数量 示例值：1
CurrentExchanges	Integer	当前vhost的exchange数量 示例值：1
CurrentConnections	Integer	当前vhost的连接数量 示例值：1
CurrentChannels	Integer	当前vhost的channel数量 示例值：1
CurrentUsers	Integer	当前vhost的用户数量 示例值：1

UserQuota

用户数配额

被如下接口引用：DescribeRabbitMQServerlessInstance。

名称	类型	描述
MaxUser	Integer	最大用户数 示例值：200
UsedUser	Integer	已用用户数 示例值：150

VirtualHostQuota

虚拟主机配额

被如下接口引用：DescribeRabbitMQServerlessInstance。

名称	类型	描述
MaxVirtualHost	Integer	最大虚拟主机数 示例值：10000
UsedVirtualHost	Integer	已经使用的虚拟主机数 示例值：10

VpcEndpointInfo

VPC接入点信息

被如下接口引用：DescribeRabbitMQServerlessInstance。

名称	类型	必选	描述
VpcId	String	是	vpc的id 示例值：vpc-xxxx
SubnetId	String	是	子网id 示例值：subnet-xxx
VpcEndpoint	String	是	vpc接入点信息 示例值：ip:port
VpcDataStreamEndpointStatus	String	否	vpc接入点状态 OFF/ON/CREATING/DELETING 示例值：OFF/ON/CREATING/DELETING

错误码

最近更新时间：2025-02-17 15:13:46

功能说明

如果返回结果中存在 Error 字段，则表示调用 API 接口失败。例如：

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

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

错误码列表

公共错误码

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

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

业务错误码

错误码	说明
AuthFailure	CAM签名/鉴权错误。
InvalidFilter	无效的过滤器
OperationDenied	操作被拒绝。
RegionError	地域错误
ResourcesSoldOut	资源售罄。