标签

产品文档

文档目录

- 产品简介
  - 产品概述
  - 产品功能
  - 使用限制
  - 支持标签的产品
- 操作指南
  - 通过标签查询资源
  - 管理标签
- 故障处理
- 日常巡检
- 常见问题
- API文档
  - 平台产品中心（tag）
    - 版本（2018-08-13）
      - API概览
      - 调用方式
        - 接口签名v1
        - 接口签名v3
        - 请求结构
        - 返回结果
        - 公共参数
      - 写接口
        - 标签关联资源
        - 创建标签
        - 标签解绑资源
        - 删除标签
        - 批量修改资源关联的标签
      - 读接口
        - 查询资源关联标签
        - 查看资源关联的标签
        - 查询标签列表
      - 数据结构
      - 错误码

产品简介
产品概述

最近更新时间: 2024-08-23 15:08:00

简介

随着科技发展，各种资源的数量和种类也随着增加，对有用的资源进行分类整理，有助于提高工作效率，所以越来越多的人喜欢使用标签来整理自己的资源文件。人们可以通过关键字、内容的相似度等因素来进行标签分类，便于以后查找和定位。这样使得标签管理发展变得日益重要。
随着云平台源数量的增加，用户管理资源的难度也随之增加。为方便用户更快速有效地查询和管理各种资源，TCE推出标签这一产品。人们可以通过标签来进行云资源的分类管理，通过预设标签来进行资源规划。标签是一些充当元数据的词和短语，用于标识和组织云资源。标签限制随资源而有所不同，但大多数最多可以有50个标签。

> **说明：**
>
> 一个标签可以对应多个资源，一个资源可以对应多个标签。

词汇解释

标签键

标签键是云提供的用于标识云上资源的一种标记，设置标签的必填项。

标签值

标签值是云提供的用于标识云上资源的一种标记，与标签键成对出现，归属于标签键，可为空值。

工作流程

1. 首先注册并登录账户，进入标签控制平台。

2. 在标签控制平台中可以进行以下两种操作：

- 将已有资源关联标签，建立已有资源列表。

- 建立标签键列表。

产品功能

最近更新时间: 2024-08-23 15:08:00

产品主要功能是查询资源和创建标签、标签授权。

查询资源

对于您在云上的已有资源，可根据资源的地域、类型以及实例来查看并进行打标签处理，查看到的资源将会以列表的形式展示。您可以根据需求来对资源命名标签键，实现对这些资源分类管理。

创建标签

创建一个标签就相当于在本地创建了一个数据库，它可以存储您在云所有资源标签信息，这样有助于您日后的资源管理。创建标签包括创建标签键和标签值。

标签授权

标签授权需要到访问管理 > 策略管理 > 新建自定义策略 > 按标签授权，去完成授权动作。

使用限制

最近更新时间: 2024-08-23 15:08:00

标签键限制

- 以 qcs:、project、项目等开头的标签键为系统预留标签键，系统预留标签键禁止创建。
- 支持UTF-8格式表示的字符、空格和数字以及特殊字符 `+ - = . _ : / @` 。
- 标签键长度 1-127个字符（采用 UTF-8 格式）。
- 标签键区分大小写。

标签值限制

- 支持UTF-8格式表示的字符、空格和数字以及特殊字符 `+ - = . _ : / @` 。
- 标签值长度1-255个字符（采用 UTF-8 格式）。
- 标签值区分大小写。

标签数量限制

- 资源维度：一个资源最多50个不同的key。
- 标签维度：
  - 单个用户最多1000个不同的key。
  - 一个key最多有1000个value。

版权所有：尚航云_V1

支持标签的产品

最近更新时间: 2024-08-23 15:08:00

计算

产品名称	资源
云服务器 CVM	SSH 密钥

容器

产品名称	资源
容器服务 TKE	容器集群

存储

产品名称	资源
对象存储 COS	存储桶
文件存储 CFS	文件系统
云硬盘 CBS	云硬盘实例

网络

产品名称	资源
负载均衡 CLB	存储桶
私有网络 VPC [1]	文件系统

> [1] 私有网络中子网、路由表、弹性网卡、安全组已支持使用标签。

数据库

产品名称	资源
云数据库 MySQL	MySQL 实例
云数据库 MongoDB	MongoDB 实例

产品名称	资源
云数据库 Redis	Redis 实例
时序数据库 CTSDB	云数据库 CTSDB 实例
分布式HTAP数据库 TBase	云数据库 TBase 实例

中间件

产品名称	资源
消息队列 CKafka	Ckafka 实例
消息队列 CMQ	队列
消息队列 CMQ	主题

应用安全

产品名称	资源
Web 应用防火墙	Web 应用防火墙

数据安全

产品名称	资源
堡垒机 BH	堡垒机
数据安全审计 DSAudit	数据安全审计
云加密机	加密机
密钥管理系统KMS	用户密钥
凭据管理系统	凭据

云智大数据平台

产品名称	资源
Elasticsearch Service	Elasticsearch 集群

域名服务

产品名称	资源
VPC域名解析	VPCDNS私有域

操作指南
通过标签查询资源

最近更新时间: 2024-08-23 15:08:00

本文档介绍如何通过标签键/值查询所属资源并导出列表，以及编辑标签值的详细操作。

查询标签

1. 登录标签控制台。

2. 在左侧导航栏中，单击**资源标签**，进入资源标签页面。

3. 在资源标签页面，选择以下信息设置筛选规则。

- 地域：需要查询资源所属地域。
- 资源类型：需要查询资源所属类型，仅支持标签的产品，详情请参见 支持标签的产品。
- 标签键/值：需要查询资源所属标签键/值，标签值可为空，支持多选。

4. 单击**查询**，页面下方会以列表形式展示出对应的资源，完成资源查询操作。

导出标签

勾选指定标签前的复选框，在编辑标签值的右侧选择 ，导出标签列表。

编辑标签值

- 批量编辑

  i. 勾选指定标签前的复选框，单击**编辑标签值**，打开**编辑标签**对话框。

  ii. 在**编辑标签**对话框中，编辑标签值。

  iii. 单击**确定**。

- 逐个编辑

选择指定标签，单击标签值右侧的 ，编辑标签值。

删除标签值

- 批量编辑

  i. 勾选指定标签前的复选框，单击**编辑标签**，打开**编辑标签**对话框。

  ii. 在**编辑标签**对话框中，在指定标签值右侧单击 。

  iii. 单击**确定**。

- 逐个编辑

选择指定标签，单击标签值右侧的 ，删除标签值。

管理标签

最近更新时间: 2024-08-23 15:08:00

本文档介绍创建、导入标签的详细操作。

创建标签

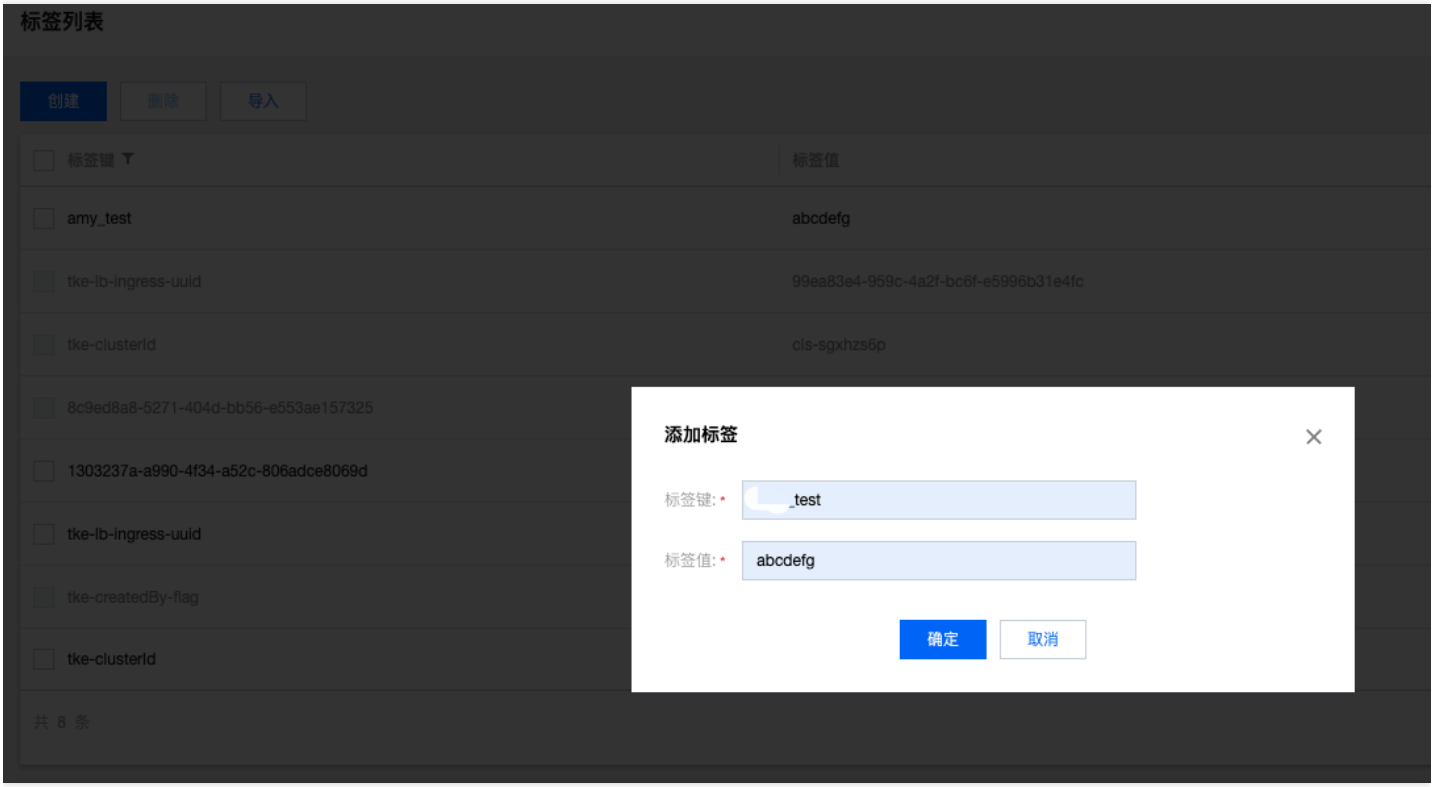
1. 登录标签控制台。

2. 在左侧导航栏中，单击**标签列表**，进入**标签列表**页面。

3. 单击**创建**，打开**添加标签**对话框。

4. 设置标签键和标签值。

5. 单击**确定**。



导入标签

1. 登录标签控制台。

2. 在左侧导航栏中，单击**标签列表**，进入**标签列表**页面。

3. 单击**导入**，打开**批量添加**对话框。

4. 单击**点此下载**，下载模版，并在本地添加标签键和标签值。

5. 单击**选择文件**，选择已配置好的模版文件。

6. 单击**确认**。

故障处理

最近更新时间: 2024-08-23 15:08:00

查询异常

故障现象

无法在前端查看到标签列表。

故障影响

已创建标签或资源已绑定标签无法查询。

故障处理

进入tcloud-tcenter-platform-tag容器:

```
ps -ef |grep -i tag,查看进程是否正常。不正常可以看下/data/release/swoole_tag/application/log.txt，这里会记录下进程退出的原因

如果进程正常可以查看下 /data/release/swoole_tag/application/logs/req下面的请求日志，以及/data/release/swoole_tag/application/logs/detail的详细日志
```

创建异常

故障现象

创建或绑定标签失败。

故障影响

无法创建标签或者无法对资源进行绑定。

故障处理

```
ps -ef |grep -i wtag,查看进程是否正常。不正常可以看下/data/release/swoole_tag_write/application/log.txt，这里会记录下进程退出的原因
如果进程正常可以查看下 /data/release/swoole_tag_write/application/logs/req下面的请求日志，以及 /data/release/swoole_tag_write/application/logs/detail的详细日志
```

日常巡检

最近更新时间: 2024-08-23 15:08:00

服务可用性性检测

通过对服务的域名+端口进行curl，验证标签服务服务是否正常，例如：

1. kg tcloud-tcenter-platform-tag，找到容器IP，比如: 192.168.241.175 curl -v -d '' http://192.168.241.175:50030 header状态码返回 200，则表明访问服务正常。

2. tcloud-tcenter-platform-wtag，找到容器IP，比如：192.168.239.227 curl -v -d '' http://192.168.239.227:50031 header状态码返回 200，则表明访问服务正常。

服务健康性检测

登录租户端，进入到标签控制台能够正常拉出标签，以及对资源进行绑定标签操作：

1. 进入标签控制台，标签列表能够创建成功标签。

2. 进入资源标签，选择业务资源，然后进行标签绑定，能够看到资源已经绑定标签。

日常维护

日志会自动清理无需手动删除。

常见问题

最近更新时间: 2024-08-23 15:08:00

什么是标签？

标签云提供的用于标识云上资源的标记，是一个键值对（Key-Value）。您可以根据各种维度（例如业务、用途、负责人等）使用标签对云服务器资源进行分类管理，也可以通过标签非常方便地筛选过滤出对应的资源。

如何通过标签键筛选资源？

1. 登录标签控制台。

2. 在左侧导航栏中单击**标签列表**，进入标签列表页面。

3. 单击列表顶部的**标签键**，勾选需要的标签，单击**确定**完成筛选。

如何通过标签给资源授权？

请参见 按标签授权。

如何给特定资源添加多个标签？

请参见 编辑特定资源标签，在第4步添加多个标签键/值即可。

API文档
平台产品中心（tag）
版本（2018-08-13）
API概览

最近更新时间: 2024-09-03 18:49:41

API版本

V3

写接口

接口名称	接口功能
AddResourceTag	标签关联资源
CreateTag	创建标签
DeleteResourceTag	标签解绑资源
DeleteTag	删除标签
ModifyResourceTags	批量修改资源关联的标签

读接口

接口名称	接口功能
DescribeResourceTags	查询资源关联标签
DescribeResourceTagsByResourceIds	查看资源关联的标签
DescribeTags	查询标签列表

调用方式
接口签名v1

最近更新时间: 2024-09-03 18:49:41

tcecloud 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tcecloud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，点击【新建】即可以创建一对SecretId/SecretKey

注意：开发商帐号最多可以拥有两对 SecretId / SecretKey。

2. 生成签名串

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

假设用户的 SecretId 和 SecretKey 分别是：

- SecretId: AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE
- SecretKey: Gu5t9xGARNpq86cd98joQYCN3EXAMPLE

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

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

参数名称	中文	参数值
Action	方法名	DescribeInstances
SecretId	密钥Id	AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE
Timestamp	当前时间戳	1465185768
Nonce	随机正整数	11886
Region	实例所在区域	ap-guangzhou

参数名称	中文	参数值
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

2.1. 对参数排序

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

```
{
'Action' : 'DescribeInstances',
'InstanceIds.0' : 'ins-09dx96dg',
'Limit' : 20,
'Nonce' : 11886,
'Offset' : 0,
'Region' : 'ap-guangzhou',
'SecretId' : 'AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE',
'Timestamp' : 1465185768,
'Version': '2017-03-12',
}
```

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

2.2. 拼接请求字符串

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

然后将格式化后的各个参数用"&"拼接在一起，最终生成的请求字符串为:

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

2.3. 拼接签名原文字符串

此步骤生成签名原文字符串。 签名原文字符串由以下几个参数构成:

1. 请求方法: 支持 POST 和 GET 方式，这里使用 GET 请求，注意方法为全大写。
2. 请求主机:查看实例列表(DescribeInstances)的请求域名为：cvm.cloud.sunhongs.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.cloud.sunhongs.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Timestamp=1465185768&Version=2017-03-12
```

2.4. 生成签名串

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

具体代码如下，以 PHP 语言为例:

```
$secretKey = 'Gu5t9xGARNpq86cd98joQYCN3EXAMPLE';
$srcStr = 'GETcvm.cloud.sunhongs.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Timestamp=1465185768&Version=2017-03-12';
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));
echo $signStr;
```

最终得到的签名串为:

```
EliP9YW3pW28FpsEdkXt/+WcGeI=
```

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

3. 签名串编码

生成的签名串并不能直接作为请求参数，需要对其进行 URL 编码。

如上一步生成的签名串为 EliP9YW3pW28FpsEdkXt/+WcGeI= ，最终得到的签名串请求参数（Signature）为：EliP9YW3pW28FpsEdkXt%2f%2bWcGeI%3d，它将用于生成最终的请求 URL。

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

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

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

4. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理

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

错误代码	错误描述
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tcecloud SDK 3.0 ，SDK 封装了签名的过程，开发时只关注产品提供的具体接口即可。详细信息参见 SDK 中心。当前支持的编程语言有：

- Python
- Java
- PHP
- Go
- JavaScript
- .NET

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

最终输出的 url 可能为： https://cvm.cloud.sunhongs.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Signature=EliP9YW3pW28FpsEdkXt%2F%2BWcGeI%3D&Timestamp=1465185768&Version=2017-03-12

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

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

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

Java

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

public class TceCloudAPIDemo {
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.cloud.sunhongs.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.cloud.sunhongs.com/?");
// 实际请求的url中对参数顺序没有要求
for (String k : params.keySet()) {
// 需要对请求串进行urlencode，由于key都是英文字母，故此处仅对其value进行urlencode
url.append(k).append("=").append(URLEncoder.encode(params.get(k).toString(), CHARSET)).append("&");
}
return url.toString().substring(0, url.length() - 1);
}

public static void main(String[] args) throws Exception {
TreeMap<String, Object> params = new TreeMap<String, Object>(); // TreeMap可以自动排序
// 实际调用时应当使用随机数，例如：params.put("Nonce", new Random().nextInt(java.lang.Integer.MAX_VALUE));
params.put("Nonce", 11886); // 公共参数
// 实际调用时应当使用系统当前时间，例如： params.put("Timestamp", System.currentTimeMillis() / 1000);
params.put("Timestamp", 1465185768); // 公共参数
params.put("SecretId", "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"); // 公共参数
params.put("Action", "DescribeInstances"); // 公共参数
params.put("Version", "2017-03-12"); // 公共参数
params.put("Region", "ap-guangzhou"); // 公共参数
params.put("Limit", 20); // 业务参数
params.put("Offset", 0); // 业务参数
params.put("InstanceIds.0", "ins-09dx96dg"); // 业务参数
params.put("Signature", sign(getStringToSign(params), "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE", "HmacSHA1")); // 公共参数
System.out.println(getUrl(params));
}
}
```

Python

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

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

import requests
```

```
secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"

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

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

if __name__ == '__main__':
endpoint = "cvm.cloud.sunhongs.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)
```

接口签名v3

最近更新时间: 2024-09-03 18:49:41

tcecloud 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tcecloud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，点击【新建】即可以创建一对SecretId/SecretKey

注意：开发商帐号最多可以拥有两对 SecretId / SecretKey。

2. TC3-HMAC-SHA256 签名方法

注意：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 格式默认所有业务接口均支持，multipart 格式只有特定业务接口支持，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。

下面以云服务器查询广州区实例列表作为例子，分步骤介绍签名的计算过程。我们仅用到了查询实例列表的两个参数：Limit 和 Offset，使用 GET 方法调用。

假设用户的 SecretId 和 SecretKey 分别是：AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE 和 Gu5t9xGARNpq86cd98joQYCN3EXAMPLE

2.1. 拼接规范请求串

按如下格式拼接规范请求串（CanonicalRequest）：

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

- HTTPRequestMethod：HTTP 请求方法（GET、POST ），本示例中为 GET；

- 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，本示例取值为：Limit=10&Offset=0。注意：CanonicalQueryString 需要经过 URL 编码。
- CanonicalHeaders：参与签名的头部信息，至少包含 host 和 content-type 两个头部，也可加入自定义的头部参与签名以提高自身请求的唯一性和安全性。拼接规则：1）头部 key 和 value 统一转成小写，并去掉首尾空格，按照 key:value\n 格式拼接；2）多个头部，按照头部 key（小写）的字典排序进行拼接。此例中为：`content-type:application/x-www-form-urlencoded\nhost:cvm.cloud.sunhongs.com\n`
- SignedHeaders：参与签名的头部信息，说明此次请求有哪些头部参与了签名，和 CanonicalHeaders 包含的头部内容是一一对应的。content-type 和 host 为必选头部。拼接规则：1）头部 key 统一转成小写；2）多个头部 key（小写）按照字典排序进行拼接，并且以分号（;）分隔。此例中为：`content-type;host`
- HashedRequestPayload：请求正文的哈希值，计算方法为 Lowercase(HexEncode(Hash.SHA256(RequestPayload)))，对 HTTP 请求整个正文 payload 做 SHA256 哈希，然后十六进制编码，最后编码串转换成小写字母。注意：对于 GET 请求，RequestPayload 固定为空字符串，对于 POST 请求，RequestPayload 即为 HTTP 请求正文 payload。

根据以上规则，示例中得到的规范请求串如下（为了展示清晰，\n 换行符通过另起打印新的一行替代）：

```
GET
/
Limit=10&Offset=0
content-type:application/x-www-form-urlencoded
host:cvm.cloud.sunhongs.com

content-type;host
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

2.2. 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
```

- Algorithm：签名算法，目前固定为 TC3-HMAC-SHA256；
- RequestTimestamp：请求时间戳，即请求头部的 X-TC-Timestamp 取值，如上示例请求为 1539084154；
- CredentialScope：凭证范围，格式为 Date/service/tc3_request，包含日期、所请求的服务和终止字符串（tc3_request）。**Date 为 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致**；service 为产品名，必须与调用的产品域名一致，例如 cvm。如上示例请求，取值为 2018-10-09/cvm/tc3_request；
- HashedCanonicalRequest：前述步骤拼接所得规范请求串的哈希值，计算方法为 Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))。

注意：

> 1. Date 必须从时间戳 X-TC-Timestamp 计算得到，且时区为 UTC+0。如果加入系统本地时区信息，例如东八区，将导致白天和晚上调用成功，但是凌晨时调用必定失败。假设时间戳为 1551113065，在东八区的时间是 2019-02-26 00:44:25，但是计算得到的 Date 取 UTC+0 的日期应为 2019-02-25，而不是 2019-02-26。

> 2. Timestamp 必须是当前系统时间，且需确保系统时间和标准时间是同步的，如果相差超过五分钟则必定失败。如果长时间不和标准时间同步，可能导致运行一段时间后，请求必定失败（返回签名过期错误）。

根据以上规则，示例中得到的待签名字符串如下（为了展示清晰，\n 换行符通过另起打印新的一行替代）：

```
TC3-HMAC-SHA256
1539084154
2018-10-09/cvm/tc3_request
91c9c192c14460df6c1ffc69e34e6c5e90708de2a6d282cccf957dbf1aa7f3a7
```

2.3. 计算签名

1）计算派生签名密钥，伪代码如下

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

- SecretKey：原始的 SecretKey；
- Date：即 Credential 中的 Date 字段信息，如上示例，为2018-10-09；
- Service：即 Credential 中的 Service 字段信息，如上示例，为 cvm；

2）计算签名，伪代码如下

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

- SecretSigning：即以上计算得到的派生签名密钥；
- StringToSign：即步骤2计算得到的待签名字符串；

2.4. 拼接 Authorization

按如下格式拼接 Authorization：

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

- Algorithm：签名方法，固定为 TC3-HMAC-SHA256；
- SecretId：密钥对中的 SecretId；
- CredentialScope：见上文，凭证范围；
- SignedHeaders：见上文，参与签名的头部信息；
- Signature：签名值

根据以上规则，示例中得到的值为：

```
TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=5
da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
```

最终完整的调用信息如下：

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

Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE/2018-10-09/cvm/tc3_request, SignedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
Content-Type: application/x-www-form-urlencoded
Host: cvm.cloud.sunhongs.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1539084154
X-TC-Region: ap-guangzhou
```

3. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理

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

4. 签名演示

Java

```
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Map;
import java.util.TimeZone;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.net.ssl.HttpsURLConnection;
import javax.xml.bind.DatatypeConverter;

import org.apache.commons.codec.digest.DigestUtils;

public class TceCloudAPITC3Demo {
private final static String CHARSET = "UTF-8";
```

```
private final static String ENDPOINT = "cvm.cloud.sunhongs.com";
private final static String PATH = "/";
private final static String SECRET_ID = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE";
private final static String SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE";
private final static String CT_X_WWW_FORM_URLENCODED = "application/x-www-form-urlencoded";
private final static String CT_JSON = "application/json";
private final static String CT_FORM_DATA = "multipart/form-data";

public static byte[] sign256(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(CHARSET));
}

public static void main(String[] args) throws Exception {
String service = "cvm";
String host = "cvm.cloud.sunhongs.com";
String region = "ap-guangzhou";
String action = "DescribeInstances";
String version = "2017-03-12";
String algorithm = "TC3-HMAC-SHA256";
String timestamp = "1539084154";
//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 = "GET";
String canonicalUri = "/";
String canonicalQueryString = "Limit=10&Offset=0";
String canonicalHeaders = "content-type:application/x-www-form-urlencoded\n" + "host:" + host + "\n";
String signedHeaders = "content-type;host";
String hashedRequestPayload = DigestUtils.sha256Hex("");
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 = DigestUtils.sha256Hex(canonicalRequest.getBytes(CHARSET));
String stringToSign = algorithm + "\n" + timestamp + "\n" + credentialScope + "\n" + hashedCanonicalRequest;
System.out.println(stringToSign);

// ************* 步骤 3：计算签名 *************
byte[] secretDate = sign256(("TC3" + SECRET_KEY).getBytes(CHARSET), date);
byte[] secretService = sign256(secretDate, service);
byte[] secretSigning = sign256(secretService, "tc3_request");
String signature = DatatypeConverter.printHexBinary(sign256(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("Host", host);
headers.put("Content-Type", CT_X_WWW_FORM_URLENCODED);
headers.put("X-TC-Action", action);
headers.put("X-TC-Timestamp", timestamp);
headers.put("X-TC-Version", version);
headers.put("X-TC-Region", region);
}
}
```

Python

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

# 密钥参数
secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"

service = "cvm"
host = "cvm.cloud.sunhongs.com"
endpoint = "https://" + host
region = "ap-guangzhou"
action = "DescribeInstances"
version = "2017-03-12"
algorithm = "TC3-HMAC-SHA256"
timestamp = 1539084154
date = datetime.utcfromtimestamp(timestamp).strftime("%Y-%m-%d")
params = {"Limit": 10, "Offset": 0}

# ********** 步骤 1：拼接规范请求串 **********
http_request_method = "GET"
canonical_uri = "/"
canonical_querystring = "Limit=10&Offset=0"
ct = "x-www-form-urlencoded"
payload = ""
if http_request_method == "POST":
canonical_querystring = ""
ct = "json"
payload = json.dumps(params)
canonical_headers = "content-type:application/%s\nhost:%s\n" % (ct, host)
signed_headers = "content-type;host"
hashed_request_payload = hashlib.sha256(payload.encode("utf-8")).hexdigest()
canonical_request = (http_request_method + "\n" +
canonical_uri + "\n" +
canonical_querystring + "\n" +
canonical_headers + "\n" +
signed_headers + "\n" +
hashed_request_payload)
print(canonical_request)

# ********** 步骤 2：拼接待签名字符串 **********
credential_scope = date + "/" + service + "/" + "tc3_request"
```

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


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

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

# 公共参数添加到请求头部
headers = {
"Authorization": authorization,
"Host": host,
"Content-Type": "application/%s" % ct,
"X-TC-Action": action,
"X-TC-Timestamp": str(timestamp),
"X-TC-Version": version,
"X-TC-Region": region,
}
```

请求结构

最近更新时间: 2024-09-03 18:49:41

1. 服务地址

地域（Region）是指物理的数据中心的地理区域。tcecloud交付验证不同地域之间完全隔离，保证不同地域间最大程度的稳定性和容错性。为了降低访问时延、提高下载速度，建议您选择最靠近您客户的地域。

您可以通过 API接口 **查询地域列表** 查看完成的地域列表。

2. 通信协议

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

3. 请求方法

支持的 HTTP 请求方法:

- POST（推荐）
- GET

POST 请求支持的 Content-Type 类型：

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

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

4. 字符编码

均使用UTF-8编码。

返回结果

最近更新时间: 2024-09-03 18:49:41

正确返回结果

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

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

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

错误返回结果

若调用失败，其返回值示例如下为：

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

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

公共错误码 (TODO: 重复信息, 是否真的需要?)

返回结果中如果存在 Error 字段，则表示调用 API 接口失败。 Error 中的 Code 字段表示错误码，所有业务都可能出现的错误码为公共错误码，下表列出了公共错误码。

错误码	错误描述
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）。
AuthFailure.MFAFailure	MFA 错误。
AuthFailure.SecretIdNotFound	密钥不存在。
AuthFailure.SignatureExpire	签名过期。
AuthFailure.SignatureFailure	签名错误。
AuthFailure.TokenFailure	token 错误。
AuthFailure.UnauthorizedOperation	请求未 CAM 授权。
DryRunOperation	DryRun 操作，代表请求将会是成功的，只是多传了 DryRun 参数。
FailedOperation	操作失败。
InternalError	内部错误。
InvalidAction	接口不存在。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数错误。
NoSuchVersion	接口版本不存在。
RequestLimitExceeded	请求的次数超过了频率限制。
ResourceInUse	资源被占用。
ResourceInsufficient	资源不足。
ResourceNotFound	资源不存在。
ResourceUnavailable	资源不可用。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。
UnsupportedProtocol	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。

公共参数

最近更新时间: 2024-09-03 18:49:41

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

签名方法 v3

使用 TC3-HMAC-SHA256 签名方法时，公共参数需要统一放到 HTTP Header 请求头部中，如下：

参数名称	类型	必选	描述
X-TC-Action	String	是	操作的接口名称。取值参考接口文档中输入参数公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。
X-TC-Region	String	是	地域参数，用来标识希望操作哪个地域的数据。接口接受的地域取值参考接口文档中输入参数公共参数 Region 的说明。注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。
X-TC-Timestamp	Integer	是	当前 UNIX 时间戳，可记录发起 API 请求的时间。例如 1529223702。注意：如果与服务器时间相差超过5分钟，会引起签名过期错误。
X-TC-Version	String	是	操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
Authorization	String	是	HTTP 标准身份认证头部字段，例如： TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024 其中， - TC3-HMAC-SHA256：签名方法，目前固定取该值； - Credential：签名凭证，AKIDEXAMPLE 是 SecretId；Date 是 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致；service为产品名，必须与调用的产品域名一致，例如cvm； - SignedHeaders：参与签名计算的头部信息，content-type 和 host 为必选头部； - Signature：签名摘要。
X-TC-Token	String	否	临时证书所用的 Token ，需要结合临时密钥一起使用。临时密钥和 Token 需要到访问管理服务调用接口获取。长期密钥不需要 Token。

签名方法 v1

使用 HmacSHA1 和 HmacSHA256 签名方法时，公共参数需要统一放到请求串中，如下

参数名称	类型	必选	描述
Action	String	是	操作的接口名称。取值参考接口文档中输入参数公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。

参数名称	类型	必选	描述
Region	String	是	地域参数，用来标识希望操作哪个地域的数据。接口接受的地域取值参考接口文档中输入参数公共参数 Region 的说明。注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。
Timestamp	Integer	是	当前 UNIX 时间戳，可记录发起 API 请求的时间。例如1529223702，如果与当前时间相差过大，会引起签名过期错误。
Nonce	Integer	是	随机正整数，与 Timestamp 联合起来，用于防止重放攻击。
SecretId	String	是	在云API密钥上申请的标识身份的 SecretId，一个 SecretId 对应唯一的 SecretKey ，而 SecretKey 会用来生成请求签名 Signature。
Signature	String	是	请求签名，用来验证此次请求的合法性，需要用户根据实际的输入参数计算得出。具体计算方法参见接口鉴权文档。
Version	String	是	操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
SignatureMethod	String	否	签名方式，目前支持 HmacSHA256 和 HmacSHA1。只有指定此参数为 HmacSHA256 时，才使用 HmacSHA256 算法验证签名，其他情况均使用 HmacSHA1 验证签名。
Token	String	否	临时证书所用的 Token ，需要结合临时密钥一起使用。临时密钥和 Token 需要到访问管理服务调用接口获取。长期密钥不需要 Token 。

地域列表

地域（Region）是指物理的数据中心的地理区域。tcecloud交付验证不同地域之间完全隔离，保证不同地域间最大程度的稳定性和容错性。为了降低访问时延、提高下载速度，建议您选择最靠近您客户的地域。

您可以通过 API接口 **查询地域列表** 查看完成的地域列表。

写接口
标签关联资源

最近更新时间: 2024-09-03 18:49:41

1. 接口描述

接口请求域名： tag.api3.cloud.sunhongs.com。

本接口用于给标签关联资源

默认接口请求频率限制：20次/秒。

接口更新时间：2020-05-18 19:31:11。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：AddResourceTag
Version	是	否	String	公共参数，本接口取值：2018-08-13
Region	是	否	String	公共参数，本接口不需要传递此参数。
TagKey	是	否	String	标签键
TagValue	是	否	String	标签值
Resource	是	否	String	资源六段式描述

3. 输出参数

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

4. 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

错误码	描述
InvalidParameterValue.ResourceDescriptionError	Resource description error.
LimitExceeded.TagKey	The number of created tag keys reaches the limit of 1,000.
LimitExceeded.TagValue	Each tag key can have up to 1,000 values.

创建标签

最近更新时间: 2024-09-03 18:49:41

1. 接口描述

接口请求域名： tag.api3.cloud.sunhongs.com。

本接口用于创建一对标签键和标签值

默认接口请求频率限制：20次/秒。

接口更新时间：2021-05-28 20:04:47。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CreateTag
Version	是	否	String	公共参数，本接口取值：2018-08-13
Region	是	否	String	公共参数，本接口不需要传递此参数。
TagKey	是	否	String	标签键
TagValue	是	否	String	标签值

3. 输出参数

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

4. 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
LimitExceeded.TagKey	The number of created tag keys reaches the limit of 1,000.
LimitExceeded.TagValue	Each tag key can have up to 1,000 values.

错误码	描述
InvalidParameterValue.ReservedTagKey	System reserved tag keys `qcloud` , `tencent` and `project` cannot be created.
InvalidParameterValue.TagKeyCharacterIllegal	Tag key contains illegal characters.
InvalidParameterValue.TagKeyEmpty	Tag key cannot be empty.
InvalidParameterValue.TagKeyLengthExceeded	Tag key length exceeds limit.
InvalidParameterValue.TagValueCharacterIllegal	Tag value contains illegal characters.
InvalidParameterValue.TagValueLengthExceeded	Tag value length exceeds limit.
ResourceInUse.TagDuplicate	Tag already exists.

标签解绑资源

最近更新时间: 2024-09-03 18:49:41

1. 接口描述

接口请求域名： tag.api3.cloud.sunhongs.com。

本接口用于解除标签和资源的关联关系

默认接口请求频率限制：20次/秒。

接口更新时间：2020-05-18 19:41:43。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DeleteResourceTag
Version	是	否	String	公共参数，本接口取值：2018-08-13
Region	是	否	String	公共参数，本接口不需要传递此参数。
TagKey	是	否	String	标签键
Resource	是	否	String	资源六段式描述

3. 输出参数

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

4. 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
ResourceNotFound.AttachedTagKeyNotFound	The tag key associated with the resource doesn’t exist.

删除标签

最近更新时间: 2024-09-03 18:49:41

1. 接口描述

接口请求域名： tag.api3.cloud.sunhongs.com。

本接口用于删除一对标签键和标签值

默认接口请求频率限制：20次/秒。

接口更新时间：2020-05-18 19:47:49。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DeleteTag
Version	是	否	String	公共参数，本接口取值：2018-08-13
Region	是	否	String	公共参数，本接口不需要传递此参数。
TagKey	是	否	String	需要删除的标签键
TagValue	是	否	String	需要删除的标签值

3. 输出参数

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

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.TagAttachedResource	Tags associated with resources cannot be deleted.
ResourceNotFound.TagNonExist	Tag does not exist.

批量修改资源关联的标签

最近更新时间: 2024-09-03 18:49:41

1. 接口描述

接口请求域名： tag.api3.cloud.sunhongs.com。

本接口用于修改资源关联的所有标签

默认接口请求频率限制：200次/秒。

接口更新时间：2023-11-07 15:35:08。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyResourceTags
Version	是	否	String	公共参数，本接口取值：2018-08-13
Region	是	否	String	公共参数，本接口不需要传递此参数。
Resource	是	否	String	资源的六段式描述
ReplaceTags	否	否	Array of Tag	需要增加或修改的标签集合。如果Resource描述的资源未关联输入的标签键，则增加关联；若已关联，则将该资源关联的键对应的标签值修改为输入值。本接口中ReplaceTags和DeleteTags二者必须存在其一，且二者不能包含相同的标签键
DeleteTags	否	否	Array of TagKeyObject	需要解关联的标签集合。本接口中ReplaceTags和DeleteTags二者必须存在其一，且二者不能包含相同的标签键

3. 输出参数

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

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InvalidParameterValue.ResourceDescriptionError	Resource description error.
LimitExceeded.TagKey	The number of created tag keys reaches the limit of 1,000.
LimitExceeded.TagValue	Each tag key can have up to 1,000 values.
InvalidParameter.Tag	`Tag` parameter error.
InvalidParameterValue.DeleteTagsParamError	`DeleteTags` cannot contain the elements in `ReplaceTags` or `AddTags` .

读接口
查询资源关联标签

最近更新时间: 2024-09-03 18:49:41

1. 接口描述

接口请求域名： tag.api3.cloud.sunhongs.com。

查询资源关联标签

默认接口请求频率限制：20次/秒。

接口更新时间：2020-05-18 20:20:33。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeResourceTags
Version	是	否	String	公共参数，本接口取值：2018-08-13
Region	是	否	String	公共参数，本接口不需要传递此参数。
CreateUin	否	否	Uint64	创建者uin
ResourceRegion	否	否	String	资源所在地域
ServiceType	否	否	String	业务类型
ResourcePrefix	否	否	String	资源前缀
ResourceId	否	否	String	资源唯一标识
Offset	否	否	Uint64	数据偏移量，默认为 0, 必须为Limit参数的整数倍
Limit	否	否	Uint64	每页大小，默认为 15
CosResourceId	否	否	Uint64	是否是Cos的资源id

3. 输出参数

参数名称	类型	描述

参数名称	类型	描述
TotalCount	Uint64	**此参数对外不可见。** 结果总数
Offset	Uint64	**此参数对外不可见。** 数据位移偏量
Limit	Uint64	**此参数对外不可见。** 每页大小
Rows	TagResource	**此参数对外不可见。** 资源标签
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见公共错误码。

查看资源关联的标签

最近更新时间: 2024-09-03 18:49:41

1. 接口描述

接口请求域名： tag.api3.cloud.sunhongs.com。

用于查询已有资源标签键值对

默认接口请求频率限制：20次/秒。

接口更新时间：2020-05-18 20:18:33。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeResourceTagsByResourceIds
Version	是	否	String	公共参数，本接口取值：2018-08-13
Region	是	否	String	公共参数，本接口不需要传递此参数。
ServiceType	是	否	String	业务类型
ResourcePrefix	是	否	String	资源前缀
ResourceIds	是	否	Array of String	资源唯一标记
ResourceRegion	是	否	String	资源所在地域
Offset	否	否	Uint64	数据偏移量，默认为 0, 必须为Limit参数的整数倍
Limit	否	否	Uint64	每页大小，默认为 15

3. 输出参数

参数名称	类型	描述
TotalCount	Uint64	**此参数对外不可见。** 结果总数
Offset	Uint64	**此参数对外不可见。** 数据位移偏量

参数名称	类型	描述
Limit	Uint64	**此参数对外不可见。** 每页大小
Tags	TagResource	**此参数对外不可见。** 标签列表
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InvalidParameterValue.RegionInvalid	Region error.
InvalidParameterValue.ResourcePrefixInvalid	Resource prefix error.
InvalidParameterValue.ServiceTypeInvalid	Service type error.
InvalidParameterValue.UinInvalid	`Uin` parameter is invalid.
InvalidParameterValue.ResourceIdSizeInvalid	The number of resource exceeds the upper limit (50).

查询标签列表

最近更新时间: 2024-09-03 18:49:41

1. 接口描述

接口请求域名： tag.api3.cloud.sunhongs.com。

用于查询已建立的标签列表。

默认接口请求频率限制：20次/秒。

接口更新时间：2020-05-18 20:15:39。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeTags
Version	是	否	String	公共参数，本接口取值：2018-08-13
Region	是	否	String	公共参数，本接口不需要传递此参数。
TagKey	否	否	String	标签键,与标签值同时存在或同时不存在，不存在时表示查询该用户所有标签
TagValue	否	否	String	标签值,与标签键同时存在或同时不存在，不存在时表示查询该用户所有标签
Offset	否	否	Uint64	数据偏移量，默认为 0, 必须为Limit参数的整数倍
Limit	否	否	Uint64	每页大小，默认为 15
CreateUin	否	否	Uint64	创建者用户 Uin，不传或为空只将 Uin 作为条件查询
TagKeys	否	否	Array of String	标签键数组,与标签值同时存在或同时不存在，不存在时表示查询该用户所有标签,当与TagKey同时传递时只会本值
ShowProject	否	否	Uint64	是否展现项目标签

3. 输出参数

参数名称	类型	描述
TotalCount	Uint64	**此参数对外不可见。** 结果总数

参数名称	类型	描述
Offset	Uint64	**此参数对外不可见。** 数据位移偏量
Limit	Uint64	**此参数对外不可见。** 每页大小
Tags	TagWithDelete	**此参数对外不可见。** 标签列表
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InvalidParameterValue.UinInvalid	`Uin` parameter is invalid.

数据结构

最近更新时间: 2024-09-03 18:49:41

Tag

表示一个标签键值对

被如下接口引用：BatchCreateTag、ModifyResourceTags

名称	必选	允许NULL	类型	描述
TagKey	是	否	String	标签键
TagValue	是	否	String	标签值

TagKeyObject

标签键对象

被如下接口引用：ModifyResourceTags

名称	必选	允许NULL	类型	描述
TagKey	是	否	String	标签键
TagValue	否	否	String	标签值

TagWithDelete

表示一个标签键值对以及是否允许删除

被如下接口引用：DescribeTags

名称	必选	允许NULL	类型	描述
TagKey	是	否	String	标签键
TagValue	是	否	String	标签值
CanDelete	是	否	Uint64	是否可以删除

TagFilter

tag过滤数组多个是与的关系

被如下接口引用：DescribeResourcesBindTag、DescribeResourcesByTags

名称	必选	允许NULL	类型	描述
TagKey	是	否	String	标签键
TagValue	否	否	Array of String	标签值数组 多个值的话是或的关系

ResourceIdTag

资源标签键值

被如下接口引用：DescribeResourceTagsByTagKeys、DescribeResourcesByTags

名称	必选	允许NULL	类型	描述
ResourceId	是	否	String	资源唯一标识
TagKeyValues	是	否	String	标签键值对

TagResource

标签键值对以及资源ID

被如下接口引用：DescribeResourceTags、DescribeResourceTagsByResourceIds、DescribeResourcesBindTag

名称	必选	允许NULL	类型	描述
TagKey	是	否	String	标签键
TagValue	是	否	String	标签值
ResourceId	是	否	String	资源ID
TagKeyMd5	是	否	String	标签键MD5值
TagValueMd5	是	否	String	标签值MD5值
ServiceType	是	是	String	资源类型

ServiceTypeFilter

查询资源

被如下接口引用：DescribeResourcesBindTag

名称	必选	允许NULL	类型	描述
ServiceType	是	否	String	服务类型
ResourcePrefix	是	否	Array of String	资源前缀

错误码

最近更新时间: 2024-09-03 18:49:42

功能说明

如果返回结果中存在 Error 字段，则表示调用 API 接口失败。例如：

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

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

错误码列表

公共错误码

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

错误码	说明
LimitExceeded	超过配额限制。
MissingParameter	缺少参数错误。
NoSuchVersion	接口版本不存在。
RequestLimitExceeded	请求的次数超过了频率限制。
ResourceInUse	资源被占用。
ResourceInsufficient	资源不足。
ResourceNotFound	资源不存在。
ResourceUnavailable	资源不可用。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。
UnsupportedProtocol	http(s)请求协议错误，只支持 GET 和 POST 请求。
UnsupportedRegion	接口不支持所传地域。

业务错误码

错误码	说明
InvalidParameterValue.ReservedTagKey	System reserved tag keys `qcloud`, `tencent` and `project` cannot be created.
FailedOperation.TagAttachedResource	Tags associated with resources cannot be deleted.
InvalidParameterValue.ServiceTypeInvalid	Service type error.
LimitExceeded.TagValue	Each tag key can have up to 1,000 values.
InvalidParameterValue.RegionInvalid	Region error.
ResourceNotFound.TagNonExist	Tag does not exist.
InvalidParameterValue.ResourceIdSizeInvalid	The number of resource exceeds the upper limit (50).
InvalidParameterValue.TagFilters	Values specified for TagFilters are invalid.
InvalidParameterValue.TagValueCharacterIllegal	Tag value contains illegal characters.
InvalidParameterValue.TagListEmpty	The TagList parameter is not specified.
InvalidParameter.Tag	`Tag` parameter error.
InvalidParameterValue.ResourceDescriptionError	Resource description error.
InvalidParameterValue.TagValueLengthExceeded	Tag value length exceeds limit.

错误码	说明
InvalidParameterValue.TagKeyLengthExceeded	Tag key length exceeds limit.
ResourceInUse.TagDuplicate	Tag already exists.
InvalidParameterValue.TagKeyCharacterIllegal	Tag key contains illegal characters.
InvalidParameterValue.DeleteTagsParamError	`DeleteTags` cannot contain the elements in `ReplaceTags` or `AddTags` .
InvalidParameterValue.TagFiltersLengthExceeded	TagFilters does not exist or the value of TagFilters exceeds 6 characters.
ResourceNotFound.AttachedTagKeyNotFound	The tag key associated with the resource doesn’t exist.
LimitExceeded.TagKey	The number of created tag keys reaches the limit of 1,000.
InvalidParameterValue.TagKeyEmpty	Tag key cannot be empty.
InvalidParameterValue.UinInvalid	`Uin` parameter is invalid.
InvalidParameterValue.ResourcePrefixInvalid	Resource prefix error.