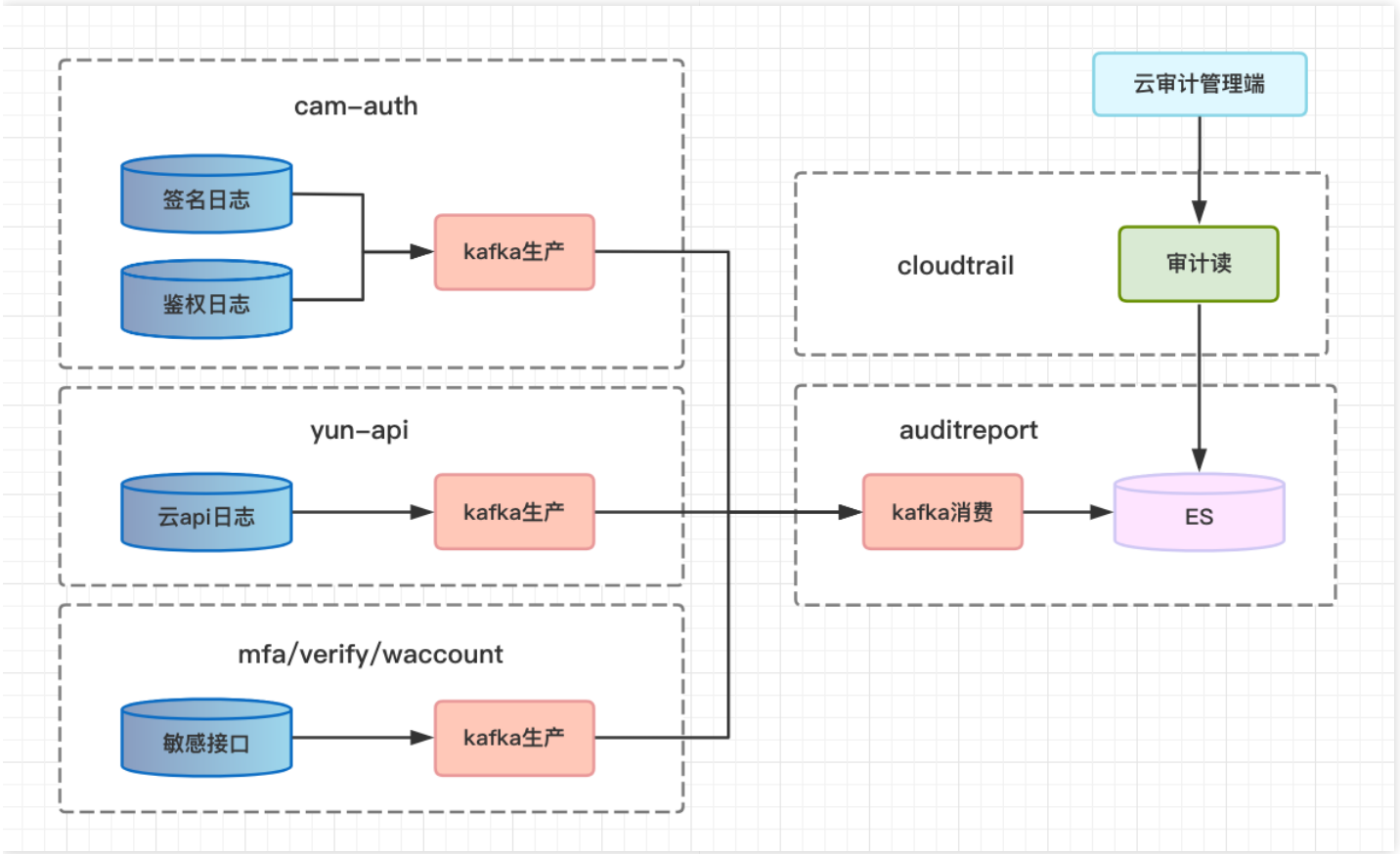
云审计

产品文档

目录

运维管理指南
架构及模块说明

云审计架构分为逻辑层和存储层。存储层使用elasticsearch 集群多副本架构，逻辑层分为读服务和写服务两个模块，读服务根据查询的条件构造es 查询语句，直接请求es server，写服务依赖kafka，数据由cam等组件发送到kafka,auditrepot组件从kafka消费数据，进行二次处理后将数据写入ES。



运维工具介绍

本部分主要介绍产品相关的重要的运维平台、工具或者命令行的使用。

运维软件

1. 解压缩软件，例如7zip，用于解压缩软件包，在工具的官方网站下载即可。
2. putty，用于在软件安装过程中在windows系统上访问各个节点，可以访问chiark主页下载。

常用脚本

1. /tce/entrypoint.sh，进程启动脚本，完成配置文件和日志目录的准备。
   执行方式： cd /tce && sh entrypoint.sh

   > **注意：**镜像启动后会自动执行该脚本，所以一般情况下无需关心。

2. /tce/healthchk.sh，健康检查脚本，对服务端口进行探测，异常的话会重启进程。

   > **注意：**镜像启动后，该脚本会定时执行。

日常巡检

服务可用性检测

根据架构图找到各服务模块，通过对服务的域名+端口进行curl，验证系统各模块是否正常，例如：

1. kg tcloud-tcenter-platform-cloudtrail
   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. kg tcloud-cloudaudit-auditreport
   进入pod里面，ps -ef查看python进程是否存在startAuth、startSignature、startSen进程，如果存在，则表明访问服务正常。

服务健康性检测

登录租户端，进入到控制台云审计页面，查看是否有数据，数据时间是否及操作是否正常，点击加载更多，是否可以看到更多合法数据。等待5分钟后，再次点击加载更多，观察是否返回异常，异常是符合预期的应答，因为数据快照过期了。

服务工作状态检查

执行ps aux | grep QC， 观察master进程和 worker 进程是否正常运行，worker进程的数据是否符合预期。

审计日志生成检查

1. 进入auditreport服务
2. cd /data/log/audit
   打开当前时间的文件，观察是否有报错，如果没有异常日志，一直有正常日志产生，，则服务正常。

故障处理
审计日志检索失败故障处理

审计日志检索失败故障处理

故障现象

进入云审计页面，看不到审计日志数据， 或根据关键字检索失败。

故障影响

较高。

故障处理

1. 进入cloudtrail服务，执行ps aux | grep QC，观察worker进程是否正常执行。如果不正常，执行sh server.sh -t restart -n QC_CLOUD_TRAIL， 进行服务重启。
2. 执行cd /data/release/cloudtrail/application/logs/sys && vim sys_{日期}.log ，查看是否有最新的 "QC_CLOUD_TRAIL_worker start" 启动日志，如果没有，那就是进程启动有问题
3. 进程启动没问题，grep lookupEvents /data/release/cloudtrail/application/logs/detail/daily_{日期}.log，观察是否有异常日志，如果有，根据response code判断是数据库异常还是es异常。进而对存储组件进行排查。
4. cloudtrail服务的response code为0，说明读服务正常，进而去排查写服务。
5. 进入auditreport服务，tail -f /data/log/audit /audit_report.logger.log，查看有没有新的正常日志生成。
6. 如果有新的正常日志，说明数据生产没问题，需要去排查es的日志，是不是出现写入的字段类型和es mapping中不兼容，导致写入失败，或者是es的其他原因，

记录数据出现异常(缺失数据)

记录数据出现异常(缺失数据)，请按以下步骤排查：

1. 进入tcloud-tcenter-cam-auth容器:

   执行tcpdump -Xnp port 9503，如果没发现数据包传递，请检查cam-auth服务是否异常，具体可查看日志/data/release/cam_swoole/log/，进行问题定位

2. 进入tcloud-tcenter-cam-log-yunapi容器:

   执行tcpdump -Xnp port 9503，查看是否收到数据包，如果没有，则为收包异常，请排查通信链路
   执行tcpdump -Xnp port 50021，观察通信是否正常，50021端口为HTTP服务，因此留意是否有关键词HTTP/1.1 200

3. 进入tcloud-cloudaudit-cloudtraillog容器

4. 观察审计日志/data/release/cloudtrail_log/application/logs/ca，如果日志不存在，请根据detail和req目录的日志进行服务诊断

5. ps aux观察filebeat进程是否存在(该进程将日志上报到ES)，可以查看/usr/local/services/qcloud_filebeat-1.0/logs执行情况

6. 根据/usr/local/services/qcloud_filebeat-1.0/filebeat.yml 配置的output.elasticsearch，使用curl方式查看es服务是否健康，比如curl 10.10.2.21:9202

如果租户端云审计页面打开异常，请排查：

1) 执行 kubectl get pod -n tce -o wide | grep tcloud-cloudaudit-cloudtrail 命令，查看结果的第六列，获取pod的IP。用curl检测健康情况，如：curl -v 192.168.254.250:50021，如果异常，可查看容器中的日志：/data/release/cloudtrail/application/logs
2) 检查ES可用性，进入tcloud-cloudaudit-cloudtraillog容器，根据/usr/local/services/qcloud_filebeat-1.0/filebeat.yml 配置的output.elasticsearch，使用curl方式查看es服务是否健康，比如curl 10.10.2.21:9202

应急预案
Elasticsearch系统故障

故障现象

Elasticsearch 系统故障，无法正常提供服务。

故障影响

审计日志读取失败或超时严重，新的审计日志数据写入成功率极低或失效，最终导致数据丢失。

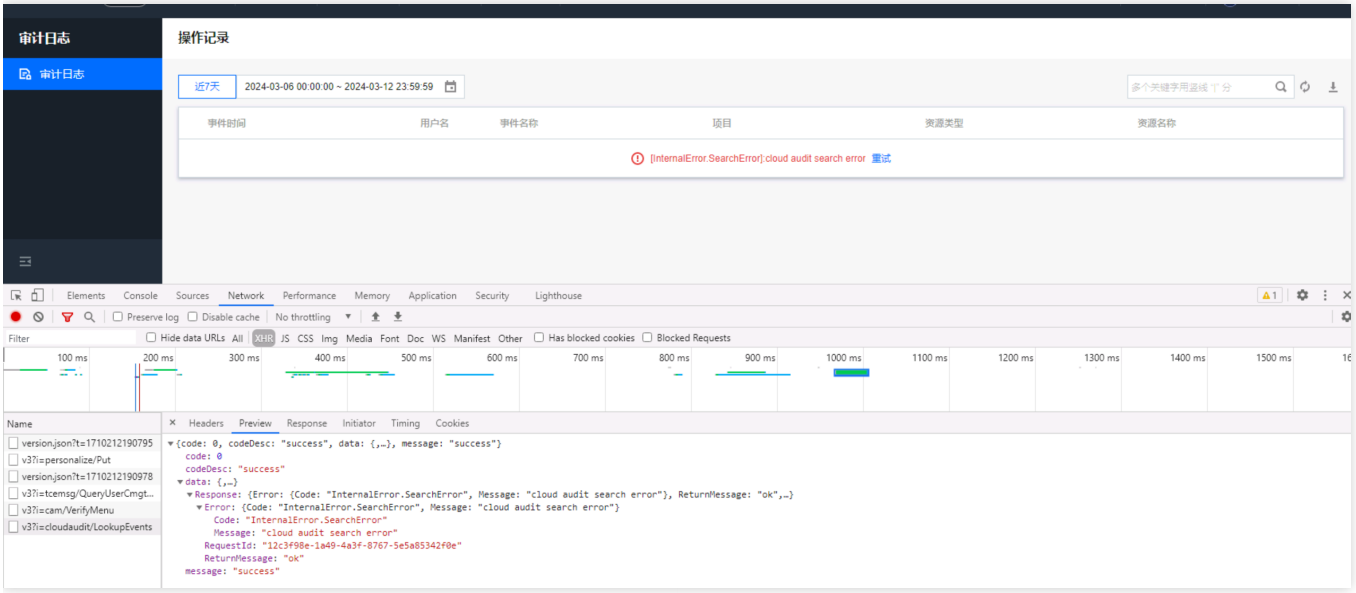
应急处理

1. 进入 auditreport 服务。
2. 先关闭健康检查，vim /tce/healthchk.sh 第一行加 exit 0。
3. cd /data/release/tcloud-cloudaudit-auditreport/scf_report 执行 bash service.sh -t stop，停止数据写入，从而减轻es server的负载。
4. 请支撑队列处理 Elasticsearch 系统问题，从控制台云审计页面，当正常查询到数据后，可以认为 Elasticsearch 恢复正常。
5. cd /data/release/tcloud-cloudaudit-auditreport/scf_report && bash service.sh -t start，重新启动 auditreport 服务进程消费kafka数据写入ES。
6. auditreport 关闭期间，数据不会被消费，数据保存在 Kafka。auditreport 重新启动，会从 Kafka 拉取数据，历史数据不会丢失。

云审计自身系统故障

故障现象

云审计页面故障，无法正常提供服务。



故障影响

审计日志无法正常查看。

应急处理

1. 后端接口直接报错场景，可能是后端服务状态异常，检查后端服务状态，确定后端pod状态都正常。
   kubectl get pod -n tce | grep cloudaudit

```
[root@tcs-10-25-2-54 tunnel_user]# kubectl get pod -n tce | grep cloudaudit
ocloud-cloudaudit-auditreport-7c757849b8-bcmtj                    1/1     Running            0          26d
ocloud-cloudaudit-auditreport-7c757849b8-hgfpv                    1/1     Running            0          312d
ocloud-cloudaudit-cloudtrail-7cb9c858b8-mrxcv                     1/1     Running            0          53d
ocloud-cloudaudit-cloudtrail-7cb9c858b8-zwz5d                     1/1     Running            0          312d
tcloud-cloudaudit-auditreport-78dbb97f69-7hjhn                    1/1     Running            0          312d
tcloud-cloudaudit-auditreport-78dbb97f69-p6n84                    1/1     Running            0          102d
tcloud-cloudaudit-cloudtrail-694ff4bbdd-kslc2                     1/1     Running            0          116d
tcloud-cloudaudit-cloudtrail-694ff4bbdd-q97p5                     1/1     Running            0          312d
[root@tcs-10-25-2-54 tunnel_user]#
```

2. 后端pod状态异常，可以快速重启pod快速恢复。

云审计操作记录无内容

故障现象

页面无明显报错，没有数据。

故障影响

审计日志无法正常查看。

应急处理

1. 页面无明显报错，没有数据，一般是支撑es或kakfa异常导致查询不到数据。

```
#进去pod内 ----租户端举例，运营端相同查询方法
kubectl exec -it -n tce tcloud-cloudaudit-cloudtrail-xxxx
cd /data/log/cloudtrail/detail
#查看最新日志，是否es相关报错
tail -f daily_2025xxxx.log

#进去pod内 ----租户端举例，运营端相同查询方法
kubectl exec -it -n tce tcloud-cloudaudit-auditreport-xxxx
cd /data/log/audit
#查看最新日志，是否有kafka相关报错
tail -f audit_report.logger.log
```

2. 检查es状态、索引。

   租户端

```
ip=`kubectl get secret -n tce es-cloudaudit -oyaml | grep ipv4 | awk -F ' ' '{print $2}' | base64 -d`
port=`kubectl get secret -n tce es-cloudaudit -oyaml | grep port | awk -F ' ' '{print $2}' | base64 -d`
user=`kubectl get secret -n tce es-cloudaudit -oyaml -oyaml | grep -w user | awk -F ' ' '{print $2}' | base64 -d`
pass=`kubectl get secret -n tce es-cloudaudit -oyaml | grep -w password | awk -F ' ' '{print $2}' | base64 -d`

#查看es集群状态是否正常
curl -u $user:$pass -s http://$ip:$port/_cluster/health?pretty
#查看es集群是否只读
curl -u $user:$pass -s http://$ip:$port/_cluster/settings | python -m json.tool | grep -i read_only
#查看索引
curl -u $user:$pass -s http://$ip:$port/_cat/indices?v | grep traillog_server_log
```

运营端

```
ip=`kubectl get secret -n tce es-ocloud-cloudaudit -oyaml | grep ipv4 | awk -F ' ' '{print $2}' | base64 -d`
port=`kubectl get secret -n tce es-ocloud-cloudaudit -oyaml | grep port | awk -F ' ' '{print $2}' | base64 -d`
user=`kubectl get secret -n tce es-ocloud-cloudaudit -oyaml -oyaml | grep -w user | awk -F ' ' '{print $2}' | base64 -d`
pass=`kubectl get secret -n tce es-ocloud-cloudaudit -oyaml | grep -w password | awk -F ' ' '{print $2}' | base64 -d`

#查看es集群状态是否正常
```

```
curl -u $user:$pass -s http://$ip:$port/_cluster/health?pretty
#查看es集群是否只读
curl -u $user:$pass -s http://$ip:$port/_cluster/settings | python -m json.tool | grep -i read_only
#查看索引
curl -u $user:$pass -s http://$ip:$port/_cat/indices?v | grep traillog_server_log
```

3. es集群正常，出现只读，一般是es集群磁盘满了，需要清理审计索引。

```
#查看索引
curl -u $user:$pass -s http://$ip:$port/_cat/indices?v | grep traillog_server_log
#清理时间旧的索引
#关闭es只读状态
curl -u $user:$pass  -H "Content-Type:application/json" -XPUT http://$ip:$port/_cluster/settings -d '{
  "persistent": {
    "cluster.blocks.read_only_allow_delete": false
  }
}'
curl -H "Content-Type: application/json" -u $user:$pass -XPUT http://$ip:$port/_all/_settings' -d '{"index":{"blocks":
{"read_only_allow_delete":null}}}

#开启批量删除模式吧
curl  -H "Content-Type: application/json" -XPUT http://$ip:$port/_cluster/settings -d '{
      "persistent": {
        "action.destructive_requires_name": false
      }
    }'
#最后删除索引
curl -XDELETE http://$ip:$port/traillog_server_log-2022*
```

- 如果es集群状态异常，需要找支撑队列处理es问题。
- 如果es集群状态正常，没有出现只读，则再检查kafka状态。

4. 检查kafka状态、topic是否创建。

    i. 参考kafka状态检查方法：

```
# 检查kafka实例pod状态
kubectl get pod -n sso | grep kafka-kafka

# 登录kafka pod内，检查审计服务的topic是否正常创建
kubectl exec -it -n sso $(kubectl get pod -n sso | grep `kubectl get sb -n tce | grep kafka-cam | grep -v export
er | awk '{print $2}'` | head -n 1 | awk '{print $1}') bash
unset JMX_PORT
# 获取zk地址
zk=$(grep '^zookeeper.connect=' /etc/kafka/server.properties | awk -F= '{print $NF}')
# 减少控制台日志输出
export KAFKA_LOG4J_OPTS=-Dlog4j.rootLogger=INFO
cd /opt/kafka_2.11-1.1.1/bin/
./kafka-topics.sh --list --zookeeper $zk | grep -E 'cloudaudit|req_production'

#对比需要创建的kafka topic
```

kubectl get sb -n tce kafka-cam -ojson | jq .spec.parameters.topicList

kubectl get sb -n tce kafka-tcenter-ocloud-cam -ojson | jq .spec.parameters.topicList

- 正常topic

```
    status: "True"
    type: InfoSynced
  message: Binding updated
  state: Ready
[root@tcs-10-27-0-25 ~]# kubectl get sb -n tce  kafka-cam  -ojson | jq .spec.parameters.topicList
[
  "auth_cloudaudit",
  "signature_cloudaudit",
  "sen_cloudaudit",
  "cloudapi_req_production_json_log"
]
[root@tcs-10-27-0-25 ~]# kubectl get sb -n tce  kafka-tcenter-ocloud-cam  -ojson | jq .spec.parameters.topicList
[
  "auth_ocloud_cloudaudit",
  "signature_ocloud_cloudaudit",
  "sen_ocloud_cloudaudit",
  "cloudapi_ocloud_req_production_json_log"
]
```

- 如果kafka状态异常，需要找支撑队列处理kafka问题。

ii. 缺少审计使用的topic，删除审计使用的sb，重建topic，再检查topic是否正常创建。

kubectl delete sb -n tce kafka-cam/kafka-tcenter-ocloud-cam

最佳实践

云审计由平台统一部署运维，具体可参考平台侧最佳实践。

节点重启

云审计容器部署，不涉及。

扩容指导

节点扩容

当系统复杂度比较高的时候，需要增加工作节点的数量。系统基于kubernetes进行容器编排及管理，可以参考开源材料《Kubernetes-集群扩容增加node节点》进行节点扩容。

swoole worker

当机器负载不搞，系统性能不足时，可以通过增加工作进程的数量来提高系统处理能力。

1. 登录机器。
2. 执行kg tagke {node id}。
3. 执行cd /data/release/swoole_tag/applicationvim product_env.php。
4. 找到WORK_NUM一行，修改对应的数字为希望扩容到的数字。
5. 重启节点。

elasticsearch集群扩容

es集群推荐使用多分片多副本的架构，当es server运算能力不足时，可以通过增加node节点进行水平扩展。可以参考开源材料《es扩容设计》。

备份恢复

通过使用多shard，多replicate架构的es集群，我们实现了数据存储的高可用性。

> 说明：
>
> 读高可用指的是多个副本情况下，某个副本出问题时不影响整个系统的读。 写高可用指的是多个副本情况下，某个副本出问题时不影响整个系统的写，通过translog来确保数据不会丢失。 集群状态的改变的高可用包含自动处理节点的加入和离开，自动同步改变的集群状态，当集群发生故障时自动切换主副shard等等来保持集群的高可用。

当副本的主节点发生故障时，es可以自动将副本切换为该 shard 的主，从而实现了数据快速恢复。

参考信息

配置文件参考

1. idc/database_config.php
    - 功能：数据库配置
    - 样例：

```
<?php class DatabaseConfig {
public static function getDbInfo(){
  $database = array( //dict结构: db_name => 配置详情
    'default' => array(
      'ip' => '127.0.0.1',
      'port' => 3306,
      'user' => 'root',
      'password' => '123456',
      'charset' => 'utf8',
      'mode' => 'async'
    ),
    'qcloudTag' => array(
      'ip' => '10.182.21.52',
      'port' => 3300,
      'user' => 'ccdb',
      'password' => 'UEDh5ZVx93IOGYCs',
      'charset' => 'utf8',
      'mode' => 'async'
    ),
    'qcloudTagCheck' => array(
      'ip' => '10.182.21.52',
      'port' => 3300,
      'user' => 'ccdb',
      'password' => 'UEDh5ZVx93IOGYCs',
      'charset' => 'utf8',
      'mode' => 'async'
    ),
    'cAuth' => array(
      'ip' => '10.182.21.52',
      'port' => 3300,
      'user' => 'ccdb',
      'password' => 'UEDh5ZVx93IOGYCs',
      'charset' => 'latin1',
      'mode' => 'async'
    ),
```

```
        );
        return $database;
    }

    /*
     * 实例名
     * 库名
     * 表明
     *
     */
    const DEFAULT_HANDLER = 'default';
    const TAG_HANDLER = 'qcloudTag';
    const TAG_CHECK_HANDLER = 'qcloudTagCheck';
    const CAUTH_HANDLER = 'cAuth';
     /*
      * db名
      *
      */
     constDEFAULT_DB = 'test';
     constTAG_DB = 'qcloudTag';
     constCAUTH_DB = 'cAuth_test';
}
```

数据库配置在物料库由工作人员统一配置，客户无需更改内容，如有变更，联系一线人员操作。

2. idc/hosts_config.php
    - 功能：下游服务配置。
    - 样例:

```
const CMSI_URL ="http://sz.cmsi.isd.com/interface.php";// 消息服务url，用于发送短息和邮件
const ACCOUNT_URL ="http://account.xxx.com:50001"; // 账户服务url
static $elk_url_module = array(
    self::ACCOUNT_URL => 'QC_ACCOUNT',
); //需要发送elk日志的下游服务
static $elk_all_http_report = true; //是否开启全量elk报告
elk_all_http_report开关打开后，会把全量的http调用日志打到elk。
```

性能指标

- 审计日志检索 pct99: 300ms。
- CPU使用率 20%。
- 内存占用率 10%。

技术指标

审计日志查询成功率 99.9%。

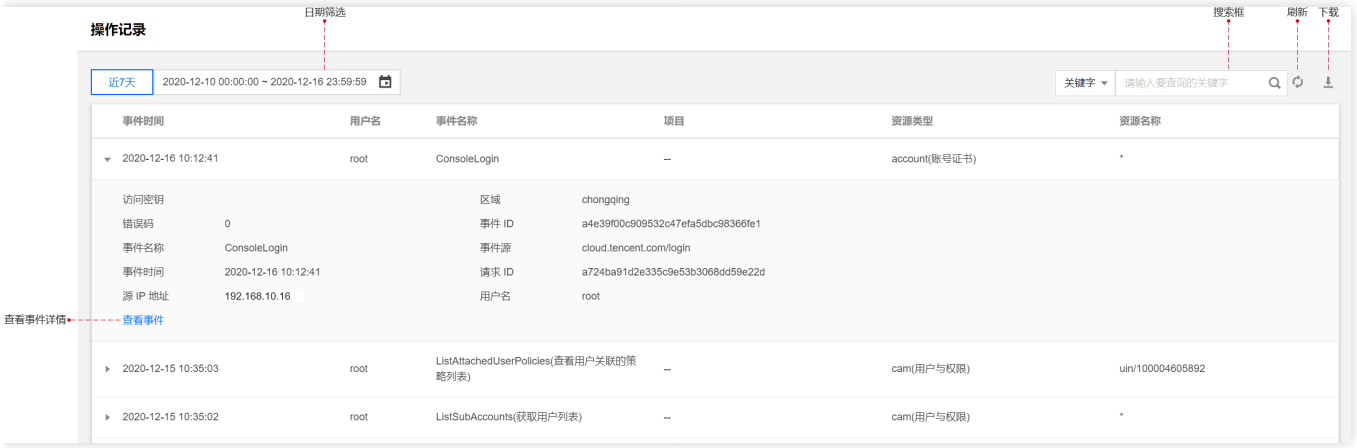
操作指南（租户端）
操作指南

1 查看操作记录

1. 登录云平台，点击【云产品】>【管理与审计】>【云审计】，进入“云审计”界面。
2. 在操作记录页面中，您可以通过关键字或者组合标签的方式进行事件搜索。其中，

- 关键字：可以在搜索框中任意输入与事件相关的关键字进行事件索索。输入后，云审计将进行关键字的匹配查询，并在列表中展示查询的结果。
- 组合标签：可以根据用户名、资源类型、项目、事件ID、事件名称、资源名、事件源或源ip地址等字段进行组合搜索。

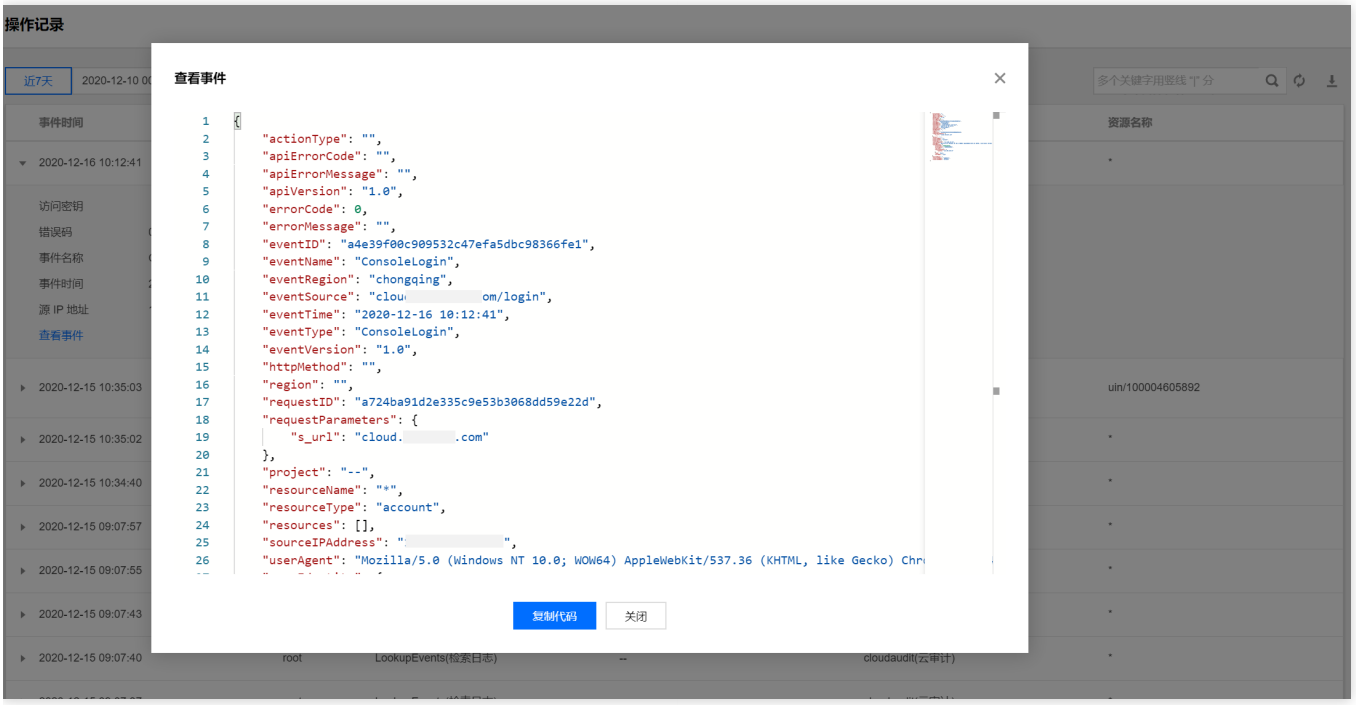
  您可以查询365天以内的操作记录，需要查询的操作记录将会以列表的形式展示出来，需要了解更多的记录 可单击【点击加载更多】。可以通过右上角的下载按钮，下载操作记录。



2 查看详情

1. 您在获取到相关的操作记录列表后，如果想更进一步了解单个操作记录，可以单击该操作记录左侧的【▼】，您就会得到此操作记录的详情，包括访问密钥、区域、错误码、事件 ID 、事件名称、事件源、事件时间、请求 ID 、源 IP 地址、用户名。同时，可以单击【查看事件】，进行了解事件的相关信息。

2. 点击“查看事件”，可以得到相关代码信息。



3 历史记录导出

1. 点击历史记录导出
2. 界面中显示的是历史的导出请求，点击申请导出。
3. 您可以将云审计的内容导出至COS中。目前支持基于时间范围进行导出。审计记录可以导出至现有的存储桶也可以导出至新建的存储桶。若导出至新建桶，需要在访问控制中创建角色：CloudAudit_QCSRole 并绑定COSFullAccess策略。


导出条件
时间范围
最近一周
最近一个月
自定义时间
存储位置
创建新的COS存储桶
是
否
所属地域 *
请选择
COS存储桶 *
请输入Bucket名称
-1255000280
仅支持小写字母、数字和 - 的组合，不能超过40字符
确定
取消

常见问题

一般性问题

什么是 CloudAudit？

CloudAudit 是一种 Web 服务，可记录在您账号上进行的活动，并将日志文件传送至您的 COS 存储桶。

CloudAudit 有哪些优势？

CloudAudit 可通过记录账号上执行的操作来提供用户活动的可见性。CloudAudit 可记录每个操作的重要信息，包括操作事件时间、用户名、事件名称、资源类型、资源名称等。这些信息能够帮助您跟踪TCloudFinanceZone资源的变更情况，帮助您解决操作性问题。

哪些人应该使用 CloudAudit？

具有以下需求的客户应该使用 CloudAudit：需要跟踪资源变更情况、回答有关用户活动的简单问题、证明合规性、进行故障排除或执行安全分析。

我可以使用哪些搜索筛选条件来查看账号活动？

您可以根据用户名、资源类型、资源名称、事件源、事件 ID 、关键词或对应操作事件时间，获取相关的账号活动。

服务

CloudAudit 支持哪些服务？

CloudAudit 目前已支持多数云服务产品，您可在 操作记录 页面的“资源事件名称”下拉列表中查看。

一个操作记录中包含了哪些信息？

一个操作记录包括访问密钥、区域、错误码、事件 ID 、事件名称、事件源、事件时间、请求 ID 、源 IP 地址、用户名。

CloudAudit 传送一个 API 调用事件需要多长时间？

一般情况下，CloudAudit 会在 API 调用后2 - 5分钟传送操作记录事件到Elasticsearch系统，并可在租户端云审计页面查看到。

其他

启用 CloudAudit 是否会影响资源的性能，或增加 API 调用的延迟？

不会。启用 CloudAudit 既不会影响资源的性能，也不会增加 API 调用的延时。

产品简介
产品概述

云审计（CloudAudit）是 一项支持对云平台账号进行监管、合规性检查、操作审核和风险审核的服务。借助 CloudAudit，您可以记录日志、持续监控并保留与整个基础设施中操作相关的账号活动。CloudAudit 提供云平台账号活动的事件历史记录，这些活动包括通过管理控制台、API 服务、命令行工具和其他服务执行的操作。这一事件历史记录可以简化安全性分析、资源更改跟踪和问题排查工作。

产品优势

为什么选择云平台审计？

优势	为什么选择云平台审计
安全性分析和故障排除	用户注册云平台账号后，可通过控制台或 API 登录云平台审计服务，避免用户信息外泄。并且在设置日志内容时，可对内容进行加密处理。借助云审计，您可以通过捕捉特定时段内您的云平台账户所发生更改的全面历史记录，发现并解决安全性和操作性问题。
简化的合规性	借助云审计，您可以自动记录和存储云平台账户中已执行操作的事件日志，从而简化合规性的审核过程，也可以更方便地搜索所有日志数据、识别不合规时间、加快事故调查速度并加快响应审核员请求的速度。
用户与资源活动的可见性	云审计可通过记录云平台相关操作和 API 调用来提高用户和资源活动的可见性。您可以识别调用云平台的用户和账户、发起调用的源 IP 地址及执行调用的时间。

与传统审计工作相比，云审计的优势

优势点	具体描述
高效性	在云审计过程中，用户操作的相关数据存储在云平台中，用户无需再把数据导入自己的办公电脑。通过云服务，将大量原本应由本机进行的计算推送到服务器端，由服务器将计算任务分配到整个云网络的空闲计算机上，迅速得到结果并返还给本地计算机，可以大大节省等待时间。
共享性	收集到用户的各项资料，采集、生成的各种数据，不再分块存储在每位用户手中，而是分类存储在同一个资源平台上；用户通过云审计平台，可以随时查阅云控制平台收集到的各项数据和资料，及时分享审计信息，避免重复劳动。
实时性	用户能够及时了解操作进展情况，并根据实际情况进行查看日志；及时了解自己的操作行为，并将操作过程中遇到的问题及时向管理者反馈：及时地了解相互的工作情况，方便实现线索、方法的共享。
监督性	通过云审计，租户端的管理者，可以实时监督每个用户的操作情况；同时运营端的管理者可以对所有租户的操作进行查询和监督，能完整地了解每个用户的操作。
合规性	通过实时，有效的记录下云平台账号内的所有操作行为，能够为问题、故障事后定位，管理部署变更追溯源头，确保运行在云平台上的业务安全合规。

产品功能

云审计主要功能：操作记录。您可以通过 API 或者控制台进行相关操作。

操作记录

操作记录功能记录最近365天的云平台 API 和云平台控制台上的所有操作。

- 操作记录列表
  您可以通过控制台查看操作记录列表，以及对应操作事件时间、用户名、事件名称、资源类型、资源名称等。

- 操作记录详情
  同时您可以获取单个操作记录详情，包括访问密钥、区域、错误码、事件 ID、事件名称、事件源、事件时间、请求 ID、源 IP 地址、用户名。

历史记录导出

云审计控制台仅支持在线查询最近365天的日志。

- 选定导出范围
  您根据关键或组合标签并配合日期选定要导出的日志
- 选定导出格式
  点击导出按钮并选择格式。目前支持：JSON和CSV两种格式

应用场景

安全分析

当用户的云账号或资源存在安全问题时，CloudAudit 所记录的日志将能帮助用户分析原因。比如，CloudAudit 会记录用户的所有账号登录操作，操作时间、源 IP 地址、是否使用多因素认证登录，这些都有详细记录，通过这些记录，用户可以判断账号是否存在安全问题。

资源变更追踪

当用户的资源出现异常变更时，CloudAudit 所记录的操作日志将能帮助用户找到原因。比如，当用户发现一台 CVM 实例停机了，用户可以通过 CloudAudit 找到相应的操作时间、源 IP 地址，以此来分析发起的停机操作。

合规性审计

如果用户的组织有多个成员，而且用户已经使用云平台的 CAM 服务来管理这些成员的身份，那么为了满足用户所在组织的合规新审计需要，用户需要获取每个成员的详细操作记录。CloudAudit 所记录的操作事件将能满足这种合规性审计需求。

使用建议

软件使用建议

- 支持记录所有云API的操作。
- 支持查询最长一年内的操作记录。

API文档
云审计（cloudaudit）
版本（2019-03-04）
API 概览

API版本

V3

云审计

接口名称	接口功能
LookupEvents	检索日志

调用方式
接口签名v1

TCloudFinanceZone 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TCloudFinanceZone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

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	shjr
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' : 'shjr',
    '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=shjr&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.finance.cloud.tencent.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.finance.cloud.tencent.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=shjr&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.finance.cloud.tencent.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=shjr&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
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TCloudFinanceZone 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
- Node

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.finance.cloud.tencent.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=shjr&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 CloudAPIDemo {
    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.finance.cloud.tencent.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.finance.cloud.tencent.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)).app
end("&");
        }
        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.Intege
r.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", "shjr"); // 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.finance.cloud.tencent.com"
    data = {
        'Action' : 'DescribeInstances',
        'InstanceIds.0' : 'ins-09dx96dg',
        'Limit' : 20,
        'Nonce' : 11886,
        'Offset' : 0,
        'Region' : 'shjr',
        '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

TCloudFinanceZone 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TCloudFinanceZone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.finance.cloud.tencent.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.finance.cloud.tencent.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=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
```

最终完整的调用信息如下：

```
https://cvm.finance.cloud.tencent.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.finance.cloud.tencent.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1539084154
X-TC-Region: shjr
```

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 CloudAPITC3Demo {
    private final static String CHARSET = "UTF-8";
    private final static String ENDPOINT = "cvm.finance.cloud.tencent.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-urlencode
d";
    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.finance.cloud.tencent.com";
        String region = "shjr";
        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" + canonicalQueryStri
ng + "\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" + hashedCan
onicalRequest;
        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)).toLow
erCase();
        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.finance.cloud.tencent.com"
endpoint = "https://" + host
region = "shjr"
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

1. 服务地址

地域（Region）是指物理的数据中心的地理区域。TCloudFinanceZone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

TCloudFinanceZone 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

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 cor
rect."
        },
        "RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
    }
}
```

- Error 的出现代表着该请求调用失败。Error 字段连同其内部的 Code 和 Message 字段在调用失败时是必定返回的。
- Code 表示具体出错的错误码，当请求出错时可以先根据该错误码在公共错误码和当前接口对应的错误码列表里面查找对应原因和解决方案。

- Message 显示出了这个错误发生的具体原因，随着业务发展或体验优化，此文本可能会经常保持变更或更新，用户不应依赖这个返回值。
- RequestId 用于一个 API 请求的唯一标识，如果 API 出现异常，可以联系我们，并提供该 ID 来解决问题。

公共错误码

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

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

错误码	错误描述
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

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。
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。

参数名称	类型	必选	描述
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。TCloudFinanceZone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。

云审计
检索日志

1. 接口描述

接口请求域名： cloudaudit.api3.finance.cloud.tencent.com。

检索日志接口

默认接口请求频率限制：20次/秒。

接口更新时间：2025-03-11 15:35:38。

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：LookupEvents
Version	是	否	String	公共参数，本接口取值：2019-03-04
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
StartTime	是	否	Uint64	查询开始时间 示例值： 1729094400000
EndTime	是	否	Uint64	查询结束时间 示例值： 1729094400000
MaxResults	否	否	Uint64	每次最大查询数量 示例值： 10
NextToken	否	否	String	查询分页 示例值： 82bf3ccad18b0db62ea4f9790b969833
OwnerUin	否	否	String	查询用户uin 示例值： 110000000023

参数名称	必选	允许NULL	类型	描述
LookupAttributes	否	否	Array of Attr	查询属性细节 示例值：查看
LookupType	否	否	String	搜索类型 示例值： keyValue
ContentValue	否	否	String	全局内容模糊查询 示例值： 100

3. 输出参数

参数名称	类型	描述
Events	Array of Events	事件 示例值：查看
NextToken	String	下一页token 示例值： 82bf3ccad18b0db62ea4f9790b969833
ListOver	Bool	是否最后一页 示例值： false
ReturnMessage	String	ReturnMessage 示例值： ok
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。

数据结构

Resources

资源内容

被如下接口引用：LookupEvents

名称	必选	允许NULL	类型	描述
ResourceName	是	否	String	资源名字 示例值： 云审计
ResourceType	是	否	String	资源类型 示例值： cloudaudit
ResourceRegion	否	否	String	资源地域 示例值： mr
ResourceCnName	否	否	String	资源中文名 示例值： 云审计
ResourceEnName	否	否	String	服务英文名 示例值： Cloud Audit
ResourceTypeName	否	否	String	服务类型 示例值： 云审计

ExportList

导出任务

被如下接口引用：GetExporLoggingList

名称	必选	允许NULL	类型	描述
Id	否	否	Uint64	ID 示例值： 15
OwnerUin	否	否	Uint64	用户ID 示例值： 110000000023
CosBucket	否	否	String	存储桶 示例值： allen3-1255000309
CosRegion	否	否	String	地域 示例值： mr
Status	否	否	Uint64	任务状态 1:待处理，2:正在处理 3:已处理 4:错误 示例值： 3
StartTime	否	否	String	开始时间 示例值： 2014-08-03 12:00:00
EndTime	否	否	String	结束时间 示例值： 2014-08-03 12:00:00
CreateTime	否	否	String	创建时间 示例值： 2014-08-03 12:00:00
UpdateTime	否	否	String	更新时间 示例值： 2014-08-03 12:00:00
SubAccountUin	否	否	Uint64	子账号ID 示例值： 110000000023
FileSize	否	是	String	文件大小 示例值： 580.934 KB
StorageType	否	否	String	存储类型。取值：cos、csp 示例值： csp
FileType	否	是	String	文件类型 示例值： json
Language	否	是	String	语言 示例值： en-US

Events

日志事件

被如下接口引用：LookupEvents

名称	必选	允许NULL	类型	描述
CloudAuditEvent	是	否	String	事件内容 示例值： {"actionType":"NonSensitiveAction","apiErrorCode":"System.Success","apiErrorMessage":"","apiVersion":"3.0","errorCode":"0","errorMessage":"","eventID":"d4af1610-0f78-84cb-1959-8165ed23b51d","eventName":"GetService","eventRegion":"on=mr","eventSource":"csp.api3.upgrade3110.fsphere.cn/","eventTime":"2024-10-23 16:56:45","eventType":"ApiCall","eventVersion":"2.0","httpMethod":"POST","region":"","requestID":"d4af1610-0f78-84cb-1959-8165ed23b51d","ukeySign":"","approver {"Action":"GetService","CosRegion":"mr","Language":"zh-CN","Nonce":"1729673805","Region":"mr","RequestClient":"CAPI_GO_SDK","SecretId":"AKID********************************","SignatureMethod":"HmacSHA256","Timestamp":"1729673805","TmpSecretId":"AKID********************************","TmpSecretKey":"taqVF0CR6gHAwhd2bh6fyOXIYcotOSOd","Version":"2020-01-07"},"project":"project1","resourceName":"","resourceType":"csp","resources": [],"sourceIPAddress":"111.206.94.145","userAgent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.0.0 Safari/537.36 Edg/130.0.0.0","userIdentity": {"principalId":"110000000023","accountId":"110000000023","secretId":"AKID********************************","sessionContext":"{"uin": 110000000023, "ownerUin": 110000000023, "appid": 1255000309, "expireTime": 1729673865, "userIp": "111.206.94.145", "randNum": 4344260345464138422, "tokenType": 1, "secretId": "AKID********************************", "system": 16}","type":"Root","userName":"110000000023(root)"},"eventZhName":"GetService","resourceCnName":"csp","resourceEnNam
EventId	是	否	String	事件id 示例值： 761afc3c-ac48-c399-ac47-e26abf2303ff
EventName	是	否	String	事件名称 示例值： LookupEvents
EventTime	是	否	String	事件时间 示例值： 2014-08-03 12:00:00
SecretId	是	否	String	SecretId 示例值： AKID********************************
ErrorCode	是	否	Uint64	错误码 示例值： 0
RequestID	是	否	String	请求ID 示例值： 761afc3c-ac48-c399-ac47-e26abf2303ff
AccountID	是	否	Uint64	账户ID 示例值： 110000000023
SourceIPAddress	是	否	String	源ip地址 示例值： 111.206.94.143
EventSource	是	否	String	事件源 示例值： csp.api3.upgrade3110.fsphere.cn/
EventRegion	是	否	String	事件域 示例值： mr
Username	是	否	String	用户名 示例值： 110000000023(root)
Resources	是	否	Resources	资源内容 示例值：查看
UkeySign	否	否	String	数字签名 示例值： 8u8i9o90
ApiErrorCode	否	否	String	yunapi错误码 示例值： System.Success
ApiErrorMessage	否	否	String	yunapi错误信息 示例值： Cloud API error.
Project	否	否	String	项目 示例值： project1
ResourceTypeCn	否	否	String	ResourceTypeCn 示例值： 账号系统-api-账号系统
ResourceRegion	否	否	String	ResourceRegion 示例值： mr
EventNameCn	否	否	String	EventNameCn 示例值： ConsoleLogin
EventZhName	否	否	String	EventZhName 示例值： QuerySwitch
ReadWriteDetail	否	是	String	接口类型 示例值： 1

Attr

查询属性细节

被如下接口引用：LookupEvents

名称	必选	允许NULL	类型	描述
AttributeKey	否	否	String	查询属性 readWriteDetail,Username,ResourceName,ResourceType,EventName 示例值： Username
AttributeValue	否	否	String	查询内容 示例值： 110000000023

错误码

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 cor
rect."
    },
    "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。

错误码	说明
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。

业务错误码

错误码	说明
InvalidParameterValue.BucketIsAlready	存储桶已存在
InvalidParameterValue.StorageTypeValueError	存储类型错误