文本内容安全

接入指引

腾讯云

文档目录

接入指引
接入准备

最近更新时间：2026-09-08 17:18:31

本文档将介绍如何接入和使用文本内容安全（Text Moderation System，TMS ）、图片内容安全（Image Moderation System，IMS）、音频内容安全（Audio Moderation System，AMS）、视频内容安全（Video Moderation System，VMS）四个产品。

编程访问配置

参考 访问控制和权限管理 提前配置好 API 权限，并获取腾讯云的 SecretID 和 SecretKey，以便对开发者的身份和权限进行校验。

控制台配置

参考 快速入门，开通 TMS 服务并领取试用包。根据您的业务需求，配置策略并获取 Biztype，这个字段将在后续测试中使用。

接入方式

支持原生 API 和 SDK 两种接入方式。
原生 API 接入需要用户根据腾讯云的签名指引，构建 Demo 生成签名进行鉴权，目的是为了验证请求者身份以及保护传输中的数据；建议用户使用 SDK 接入，SDK 封装了签名的过程，开发时只需关注产品提供的具体接口即可，接入流程较为便捷。

文本内容安全接入指引
API 接入指引

最近更新时间：2026-09-08 17:18:33

腾讯云 API 会对每个请求进行身份验证，用户使用原生 API 接入需要使用安全凭证，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，以下内容主要介绍 API 接口文档参数及 V3签名方法。

API 文档

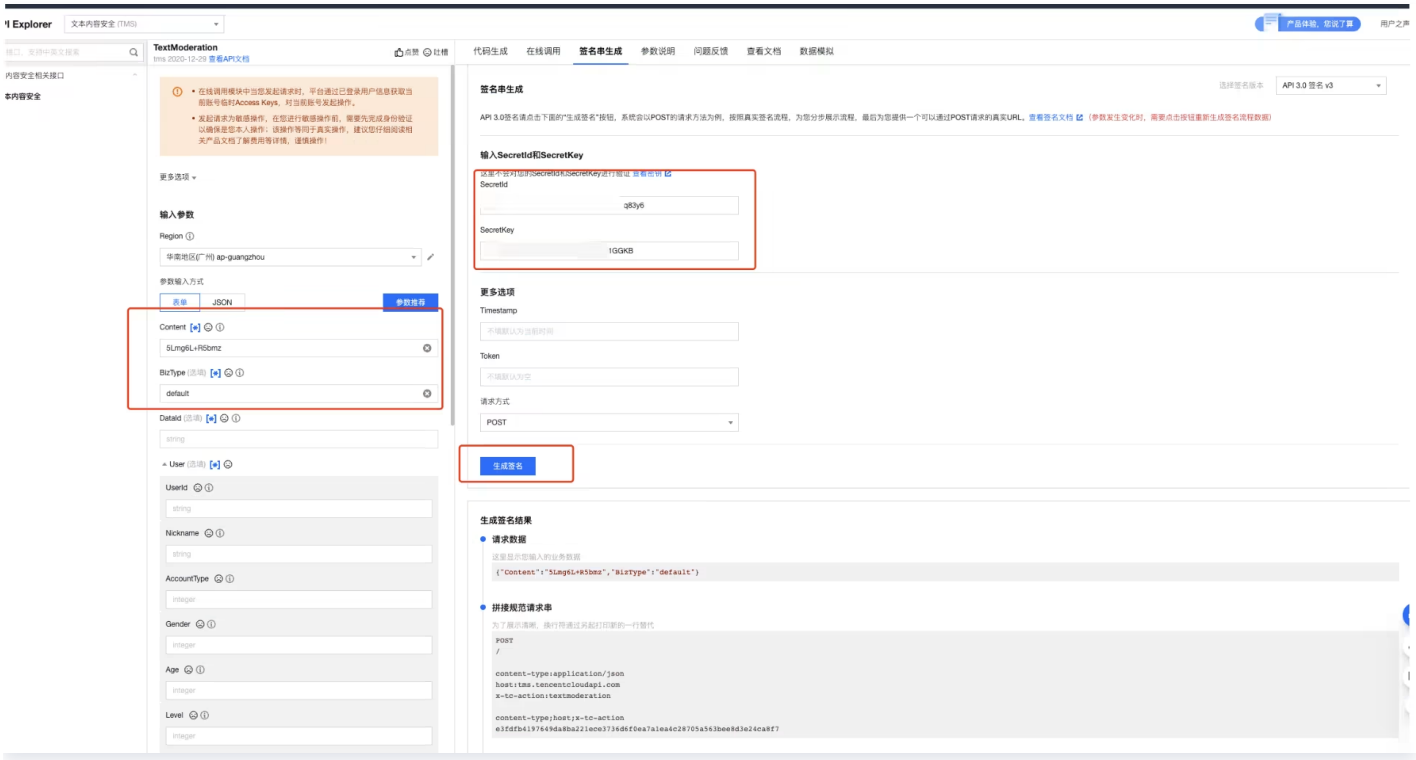
参考 文本内容安全 接口文档，了解请求 API 所需要的公共参数、域名、接口输入输出参数值。

公共参数

公共参数是用于标识用户和接口签名的参数，API 接口文档上不会展示公共参数说明，但每次请求 API 均需要携带这些参数，才能正常发起请求。

签名方法

腾讯云支持 V1、V3两种签名方式，推荐使用 V3签名方法计算签名，签名方法 v3（有时也称作 TC3-HMAC-SHA256）相比签名方法 v1更安全，支持更大的请求包，支持 POST JSON 格式，性能有一定提升。首次接触，建议使用 API Explorer 中的“签名串生成”功能，选择签名版本为“API 3.0 签名 v3”，可以生成签名过程进行验证。



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，该字段取值为 TextModeration。
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。该字段取值为2020-12-29。
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 tms；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，计算过程请参见 签名版本 v3 签名过程 。
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。

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` 协议格式。**

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，POST 请求支持 10MB 以内的请求包，签名过程详见 文档 。

示例代码

以下提供 Java、Go、Python 示例 demo，填写密钥信息、接口入参即可调用。其他语言请参考 签名方法 V3-签名演示 ，签名演示是云服务器服务作为示例，实际调用时需要根据 API 接口文档填写参数。

Java

```
import java.nio.charset.Charset;
import java.io.OutputStream;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
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 TextModeration {
    private final static Charset UTF8 = StandardCharsets.UTF_8;
    private final static String SECRET_ID = "";
    private final static String 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 = "tms";
        String host = "tms.tencentcloudapi.com";
        String region = "ap-guangzhou";
        String action = "TextModeration";
        String version = "2020-12-29";
        String algorithm = "TC3-HMAC-SHA256";
        String timestamp = String.valueOf(System.currentTimeMillis() /
1000);
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
```

```
        String date = sdf.format(new Date(Long.valueOf(timestamp +
"000")));

        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 = "{\"Content\": \"xxxx\",\"BizType\":
\"xxxx\"}";// body数据为json
        String hashedRequestPayload = sha256Hex(payload);
        String canonicalRequest = httpRequestMethod + "\n" +
canonicalUri + "\n" + canonicalQueryString + "\n"
                + canonicalHeaders + "\n" + signedHeaders + "\n" +
hashedRequestPayload;

        String credentialScope = date + "/" + service + "/" +
"tc3_request";
        String hashedCanonicalRequest = sha256Hex(canonicalRequest);
        String stringToSign = algorithm + "\n" + timestamp + "\n" +
credentialScope + "\n" + hashedCanonicalRequest;

        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();

        String authorization = algorithm + " " + "Credential=" +
SECRET_ID + "/" + credentialScope + ", "
                + "SignedHeaders=" + signedHeaders + ", " + "Signature="
+ signature;

        try {
```

```
            URL url = new URL("https://" + host);
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            connection.setRequestMethod("POST");
            connection.setRequestProperty("Authorization",
authorization);
            connection.setRequestProperty("Content-Type", CT_JSON);
            connection.setRequestProperty("Host", host);
            connection.setRequestProperty("X-TC-Action", action);
            connection.setRequestProperty("X-TC-Timestamp", timestamp);
            connection.setRequestProperty("X-TC-Version", version);
            connection.setRequestProperty("X-TC-Region", region);

            // Enable input/output streams and write the payload
            connection.setDoOutput(true);
            OutputStream os = connection.getOutputStream();
            os.write(payload.getBytes(UTF8));
            os.flush();
            os.close();

            // Get the response
            int responseCode = connection.getResponseCode();
            if (responseCode == 200) {
                BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
                String inputLine;
                StringBuilder response = new StringBuilder();

                while ((inputLine = in.readLine()) != null) {
                    response.append(inputLine);
                }
                in.close();

                // Process the response as needed
                System.out.println("Response: " + response.toString());
            } else {
                // Handle the error response
                System.out.println("Error Response Code: " +
responseCode);
            }
```

```
            connection.disconnect();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Go

```
package tms

import (
    "crypto/hmac"
    "crypto/sha256"
    "encoding/base64"
    "encoding/hex"
    "encoding/json"
    "fmt"
    "io/ioutil"
    "net/http"
    "strconv"
    "strings"
    "time"
)

type textmods struct {
    Content string
}

func sha256hexs(s string) string {
    b := sha256.Sum256([]byte(s))
    return hex.EncodeToString(b[:])
}

func hmacsha256s(s, key string) string {
    hashed := hmac.New(sha256.New, []byte(key))
    hashed.Write([]byte(s))
```

```
    return string(hashed.Sum(nil))
}

func DetectTextSingles() {
    secretId := ""                      // 密钥ID
    secretKey := ""                     // 密钥Key
    host := "tms.tencentcloudapi.com" // 接口域名
    algorithm := "TC3-HMAC-SHA256"
    service := "tms"
    version := "2020-12-29" // 版本为固定值
    action := "TextModeration"
    region := "ap-guangzhou" // 地域
    var timestamp int64 = time.Now().Unix()

    // step 1: build canonical request string
    httpRequestMethod := "POST"
    canonicalURI := "/"
    canonicalQueryString := ""
    canonicalHeaders := "content-type:application/json; charset=utf-8\n"
+ "host:" + host + "\n"
    signedHeaders := "content-type;host"
    var a = textmods{
        Content: base64.StdEncoding.EncodeToString([]byte("测试文本内
容")), // 接口入参
    }
    b, err := json.Marshal(a)
    if err != nil {
        fmt.Print(err.Error())
        return
    }
    payload := string(b)
    hashedRequestPayload := sha256hexs(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 := sha256hexs(canonicalRequest)
    string2sign := fmt.Sprintf("%s\n%d\n%s\n%s",
        algorithm,
        timestamp,
        credentialScope,
        hashedCanonicalRequest)
    // fmt.Println(string2sign)

    // step 3: sign string
    secretDate := hmacsha256s(date, "TC3"+secretKey)
    secretService := hmacsha256s(service, secretDate)
    secretSigning := hmacsha256s("tc3_request", secretService)
    signature := hex.EncodeToString([]byte(hmacsha256s(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)

    req, _ := http.NewRequest("POST", "https://"+host,
strings.NewReader(payload))
    req.Header.Set("Authorization", authorization)
    req.Header.Set("Content-Type", "application/json; charset=utf-8")
    req.Header.Set("Host", host)
    req.Header.Set("X-TC-Action", action)
    req.Header.Set("X-TC-Timestamp", strconv.FormatInt(timestamp, 10))
    req.Header.Set("X-TC-Version", version)
    req.Header.Set("X-TC-Region", region)

    resp, err := (&http.Client{}).Do(req)
```

```
    if err != nil {
        fmt.Print(err.Error())
        return
    }
    defer resp.Body.Close()

    respByte, _ := ioutil.ReadAll(resp.Body)
    fmt.Println(string(respByte))
}
```

Python

```
#!/usr/bin/env python3
# -*- coding:utf-8 -*-

import hashlib
import hmac
import json
import base64
import time
from datetime import datetime
import requests


secret_id = ""  # 密钥ID
secret_key = ""  # 密钥Key


service = "tms"
host = "tms.tencentcloudapi.com"  # 接口域名
endpoint = "https://" + host
region = "ap-guangzhou"  # 地域
version = "2020-12-29"  # 版本为固定值
algorithm = "TC3-HMAC-SHA256"


def do_action(action, params):
    timestamp = int(time.time())
    day = datetime.utcfromtimestamp(timestamp).strftime("%Y-%m-%d")
```

```
    # ************* 步骤 1：拼接规范请求串 *************
    http_request_method = "POST"
    canonical_url = "/"
    canonical_querystring = ""
    ct = "application/json; charset=utf-8"
    payload = json.dumps(params)
    canonical_headers = "content-type:%s\nhost:%s\n" % (ct, host)
    signed_headers = "content-type;host"
    hashed_request_payload = hashlib.sha256(payload.encode("utf-
8")).hexdigest()
    canonical_request = (
        http_request_method
        + "\n"
        + canonical_url
        + "\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 = day + "/" + 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)

secret_date = sign(("TC3" + secret_key).encode("utf-8"), day)
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)

headers = {
    "Authorization": authorization,
    "Content-Type": "application/json; charset=utf-8",
    "Host": host,
    "X-TC-Action": action,
    "X-TC-Timestamp": str(timestamp),
    "X-TC-Version": version,
    "X-TC-Region": region,
}

# 发送请求
resp = requests.post(url=endpoint, data=payload, headers=headers)
```

```
    return resp


# 计算签名摘要函数
def sign(key, msg):
    return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()


def image_detect(params):
    action = "TextModeration"
    resp = do_action(action, params)

    return resp.text


if __name__ == "__main__":

    params = []
    content = ""  # 接口参数
    b4content = base64.b64encode(content.encode()).decode()
    params = {"BizType": "default", "Content": b4content}
    resp = image_detect(params)
    print(resp)
```

PHP

```
<?php
//简单的后端demo
// 接口入参
$Content='';
$BizType='';

$time=time();
$Nonce=rand();
$Action='TextModeration'; // 接口名称
$Version='2020-12-29';
$Region='ap-guangzhou';
```

```php
//根据实际需求填写 SecretId 和 secretKey
$SecretId='';
$secretKey='';

//验签过程
$param["Nonce"] = $Nonce;
$param["Timestamp"] = $time;
$param["Region"] = $Region;
$param["SecretId"] = $SecretId;
$param["Action"] = $Action;
$param["Version"] = $Version;
$param["Content"] = $Content;
$param["BizType"] = $BizType;
ksort($param);
$signStr = "POSTtms.tencentcloudapi.com/?";
foreach ( $param as $key => $value ) {
    $signStr = $signStr . $key . "=" . $value . "&";
}
$signStr = substr($signStr, 0, -1);
$signature = base64_encode(hash_hmac("sha1", $signStr,$secretKey,
true));
//$signature=urlencode($signature);//文档要求必须先urlencode

//POST数据
$post_data = array(
'Action' => $Action,
'Nonce' =>$Nonce,
'SecretId'=>$SecretId,
'Signature'=>$signature,//签名
'Content' =>$Content,
'BizType' =>$BizType,
'Region' =>$Region,
'Timestamp' =>$time,
'Version' =>$Version,
);

//POST请求
$url='https://tms.tencentcloudapi.com/';
$query = http_build_query($post_data);
```

```php
$options = array(
'http' => array(
'method' => 'POST',
'header' => 'Content-type:application/x-www-form-urlencoded',
'content' => $query,
'timeout' => 15 * 60 // 超时时间（单位:s）
)
);
$context = stream_context_create($options);
$result = file_get_contents($url,false,$context);

$data_arr['code']=1;
$data_arr['result']=$result;
$data_arr['signature']=$signature;

header('content-type:application/json');
echo json_encode($data_arr);
exit();
```

签名失败

存在以下签名失败的错误码，请根据实际情况处理。

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

SDK 接入指引

最近更新时间：2026-09-08 17:18:33

推荐您使用腾讯云 API 配套的 7 种常见的编程语言 SDK，已经封装了签名和请求过程，开发时只需关注产品提供的具体接口即可。

API 文档

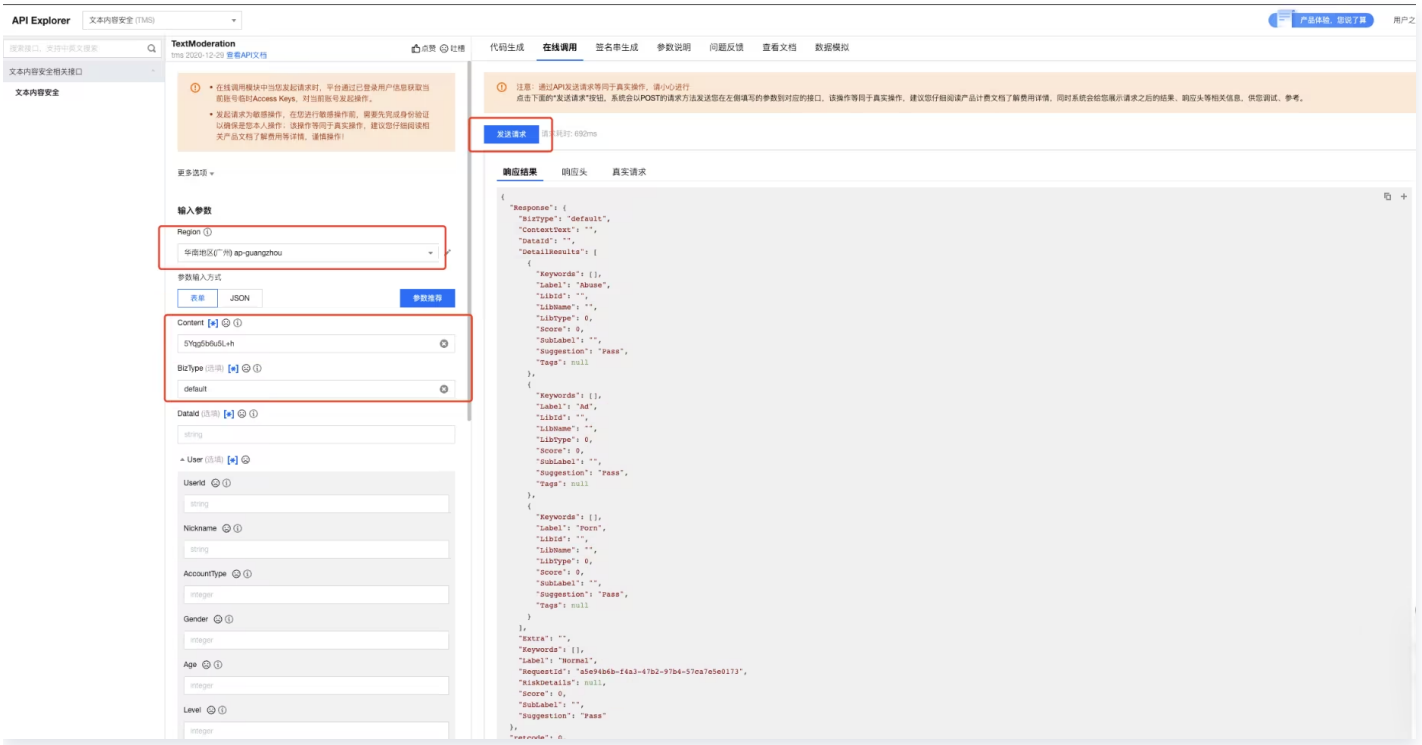
参考 文本内容安全 接口文档，了解需要使用的参数。

SDK 使用说明

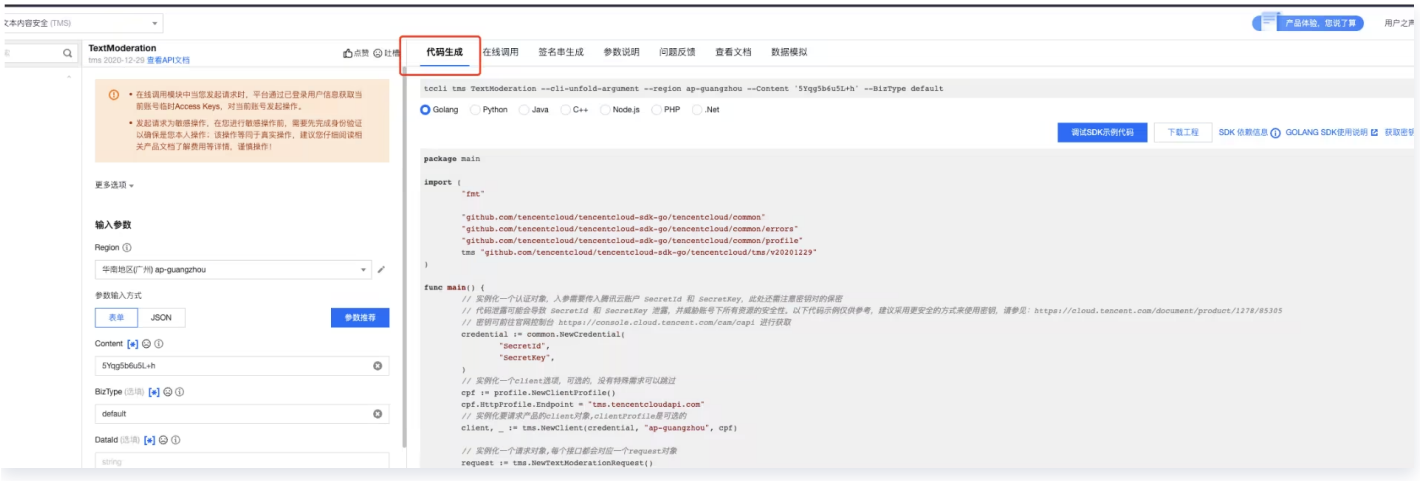
云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，均已开源，能更方便的调用 API，支持 Python 、Java 、PHP 、Go 、NodeJS 、.NET 、C++ 。
各语言 SDK 使用说明可参考 文档，介绍依赖版本以及安装过程等。

建议安装最新版本的 SDK，即可调用接口，腾讯云提供了 API Explorer 可在线调用，并且可以快速生成 SDK 调用示例代码，以下 Java、python、Golang 示例均从 API Explorer 生成，其他语言可自行调试获取。

在线调用示例



代码生成示例



Java

```
import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.profile.ClientProfile;
import com.tencentcloudapi.common.profile.HttpProfile;
import com.tencentcloudapi.common.exception.TencentCloudSDKException;
import com.tencentcloudapi.tms.v20201229.TmsClient;
import com.tencentcloudapi.tms.v20201229.models.*;

public class TextModeration
{
    public static void main(String [] args) {
        try{
            // 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处还需注意密钥对的保密
            // 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全性。以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见：https://cloud.tencent.com/document/product/1278/85305
            // 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi 进行获取
            Credential cred = new Credential("SecretId", "SecretKey");
            // 实例化一个http选项，可选的，没有特殊需求可以跳过
            HttpProfile httpProfile = new HttpProfile();
            httpProfile.setEndpoint("tms.tencentcloudapi.com");
            // 实例化一个client选项，可选的，没有特殊需求可以跳过
            ClientProfile clientProfile = new ClientProfile();
            clientProfile.setHttpProfile(httpProfile);
            // 实例化要请求产品的client对象,clientProfile是可选的
```

```
            TmsClient client = new TmsClient(cred, "ap-guangzhou",
clientProfile);
            // 实例化一个请求对象,每个接口都会对应一个request对象
            TextModerationRequest req = new TextModerationRequest();
            req.setContent("5Yqg5b6u5L+h");
            req.setBizType("default");
            // 返回的resp是一个TextModerationResponse的实例，与请求对象对应
            TextModerationResponse resp = client.TextModeration(req);
            // 输出json格式的字符串回包

System.out.println(TextModerationResponse.toJsonString(resp));
        } catch (TencentCloudSDKException e) {
            System.out.println(e.toString());
        }
    }
}
```

Go

```
package main

import (
        "fmt"

        "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common"
        "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/errors"
        "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/profile"
        tms "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/tms/v20201229"
)

func main() {
        // 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处
还需注意密钥对的保密
        // 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的
安全性。以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见：
```

```
https://cloud.tencent.com/document/product/1278/85305
        // 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi
进行获取
        credential := common.NewCredential(
                "SecretId",
                "SecretKey"
        )
        // 实例化一个client选项，可选的，没有特殊需求可以跳过
        cpf := profile.NewClientProfile()
        cpf.HttpProfile.Endpoint = "tms.tencentcloudapi.com"
        // 实例化要请求产品的client对象,clientProfile是可选的
        client, _ := tms.NewClient(credential, "ap-guangzhou", cpf)

        // 实例化一个请求对象,每个接口都会对应一个request对象
        request := tms.NewTextModerationRequest()

        request.Content = common.StringPtr("5Yqg5b6u5L+h")
        request.BizType = common.StringPtr("default")

        // 返回的resp是一个TextModerationResponse的实例，与请求对象对应
        response, err := client.TextModeration(request)
        if _, ok := err.(*errors.TencentCloudSDKError); ok {
                fmt.Printf("An API error has returned: %s", err)
                return
        }
        if err != nil {
                panic(err)
        }
        // 输出json格式的字符串回包
        fmt.Printf("%s", response.ToJsonString())
}
```

Python

```
import json
from tencentcloud.common import credential
from tencentcloud.common.profile.client_profile import ClientProfile
from tencentcloud.common.profile.http_profile import HttpProfile
```

```
from tencentcloud.common.exception.tencent_cloud_sdk_exception import
TencentCloudSDKException
from tencentcloud.tms.v20201229 import tms_client, models
try:
    # 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处还需注
意密钥对的保密
    # 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全
性。以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见:
https://cloud.tencent.com/document/product/1278/85305
    # 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi 进行获
取
    cred = credential.Credential("SecretId", "SecretKey")
    # 实例化一个http选项，可选的，没有特殊需求可以跳过
    httpProfile = HttpProfile()
    httpProfile.endpoint = "tms.tencentcloudapi.com"

    # 实例化一个client选项，可选的，没有特殊需求可以跳过
    clientProfile = ClientProfile()
    clientProfile.httpProfile = httpProfile
    # 实例化要请求产品的client对象,clientProfile是可选的
    client = tms_client.TmsClient(cred, "ap-guangzhou", clientProfile)

    # 实例化一个请求对象,每个接口都会对应一个request对象
    req = models.TextModerationRequest()
    params = {
        "Content": "5Yqg5b6u5L+h",
        "BizType": "default"
    }
    req.from_json_string(json.dumps(params))

    # 返回的resp是一个TextModerationResponse的实例，与请求对象对应
    resp = client.TextModeration(req)
    # 输出json格式的字符串回包
    print(resp.to_json_string())

except TencentCloudSDKException as err:
    print(err)
```

热点问题

SDK 是否支持设置超时时间?

SDK 具有默认的超时时间，除非必要，请勿更改默认设置。如果需要修改，可以 参考 SDK 仓库 的详细版示例。

详细版

```
import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.exception.TencentCloudSDKException;
// 导入对应产品模块的client
import com.tencentcloudapi.cvm.v20170312.CvmClient;
// 导入要请求接口对应的request response类
import com.tencentcloudapi.cvm.v20170312.models.DescribeInstancesRequest;
import com.tencentcloudapi.cvm.v20170312.models.DescribeInstancesResponse;
import com.tencentcloudapi.cvm.v20170312.models.Filter;
//导入可选配置类
import com.tencentcloudapi.common.profile.ClientProfile;
import com.tencentcloudapi.common.profile.HttpProfile;
import com.tencentcloudapi.common.profile.Language;

public class DescribeInstances {
    public static void main(String[] args) {
        try {
            // 实例化一个认证对象，入参需要传入腾讯云账户 SecretId，SecretKey。
            // 为了保护密钥安全，建议将密钥设置在环境变量中或者配置文件中，请参考本文凭证管理章节。
            // 硬编码密钥到代码中有可能随代码泄露而暴露，有安全隐患，并不推荐。
            // Credential cred = new Credential("SecretId", "SecretKey");
            Credential cred = new Credential(System.getenv("TENCENTCLOUD_SECRET_ID"), System.getenv

            // 实例化一个http选项，可选的，没有特殊需求可以跳过
            HttpProfile httpProfile = new HttpProfile();
            // 从3.0.96版本开始，单独设置 HTTP 代理
            // httpProfile.setProxyHost("真实代理ip");
            // httpProfile.setProxyPort(真实代理端口);
            httpProfile.setReqMethod("GET"); // get请求(默认为post请求)
            httpProfile.setConnTimeout(30); // 请求连接超时时间，单位为秒(默认60秒)
            httpProfile.setWriteTimeout(30);  // 设置写入超时时间，单位为秒(默认0秒)
            httpProfile.setReadTimeout(30);  // 设置读取超时时间，单位为秒(默认0秒)
            httpProfile.setEndpoint("cvm.ap-shanghai.tencentcloudapi.com"); // 指定接入地域域名(默认就近
```

多线程是否可以共用一个 Client?

SDK 目前不支持线程安全，因此需要在每个线程中创建一个 Client 来进行请求。

同时连接数量是否有限制?

目前没有限制，但是接口会限制每秒请求数（QPS）的并发数，文本接口默认为100QPS。建议客户端也控制并发数，因为太多线程可能超出用户机器的处理能力，导致连接失败。

接入后验证

最近更新时间：2026-09-08 17:18:33

当您通过上述步骤完成 API 接入后，可以通过如下方式验证文本内容检测服务是否接入成功。

接入成功

若 Response 回参没有“Error”字段且正常返回业务参数，则说明 TMS 接口接入成功，示例如下所示：


```
{
  "Response": {
    "DataId": "123",
    "Extra": "xx",
    "BizType": "0",
    "RiskDetails": [
      {
        "Level": 2,
        "Label": "RiskAccount"
      }
    ],
    "DetailResults": [
      {
        "LibName": "Porn",
        "Score": 72,
        "Label": "Porn",
        "LibId": "12",
        "Suggestion": "Review",
        "Keywords": [
          "色情"
        ],
        "LibType": 0
      },
      {
        "LibName": "Porn",
        "Score": 0,
        "Label": "",
        "LibId": "1",
        "Suggestion": "Block",
        "Keywords": [
```

```
            "色情"
          ],
          "LibType": 2
        }
    ],
    "Label": "Ad",
    "Score": 87,
    "RequestId": "x2123-123123-123",
    "Suggestion": "Block",
    "Keywords": [
      "加我好友，给你发优惠券"
    ]
  }
}
```

接入失败

若 Response 回参显示“Error”字段，则说明接入失败，以下为 signature 校验失败的示例：

```
{
    "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"
    }
}
```

其中 Error 中的 Code 表示错误码，Message 表示错误的具体信息，详情请参见 错误码 。
如您仍无法定位接入错误的原因，可以 提交工单 联系技术工程师，我们会7*24小时响应和解决您的问题。

图片内容安全接入指引
API接入指引

最近更新时间：2026-09-08 17:18:33

腾讯云 API 会对每个请求进行身份验证，用户使用原生 API 接入需要使用安全凭证，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，以下内容主要介绍 API 接口文档参数以及 V3签名方法。

API 文档

参考 图片内容安全 接口文档，了解请求 API 所需要的公共参数、域名、接口输入输出参数值。

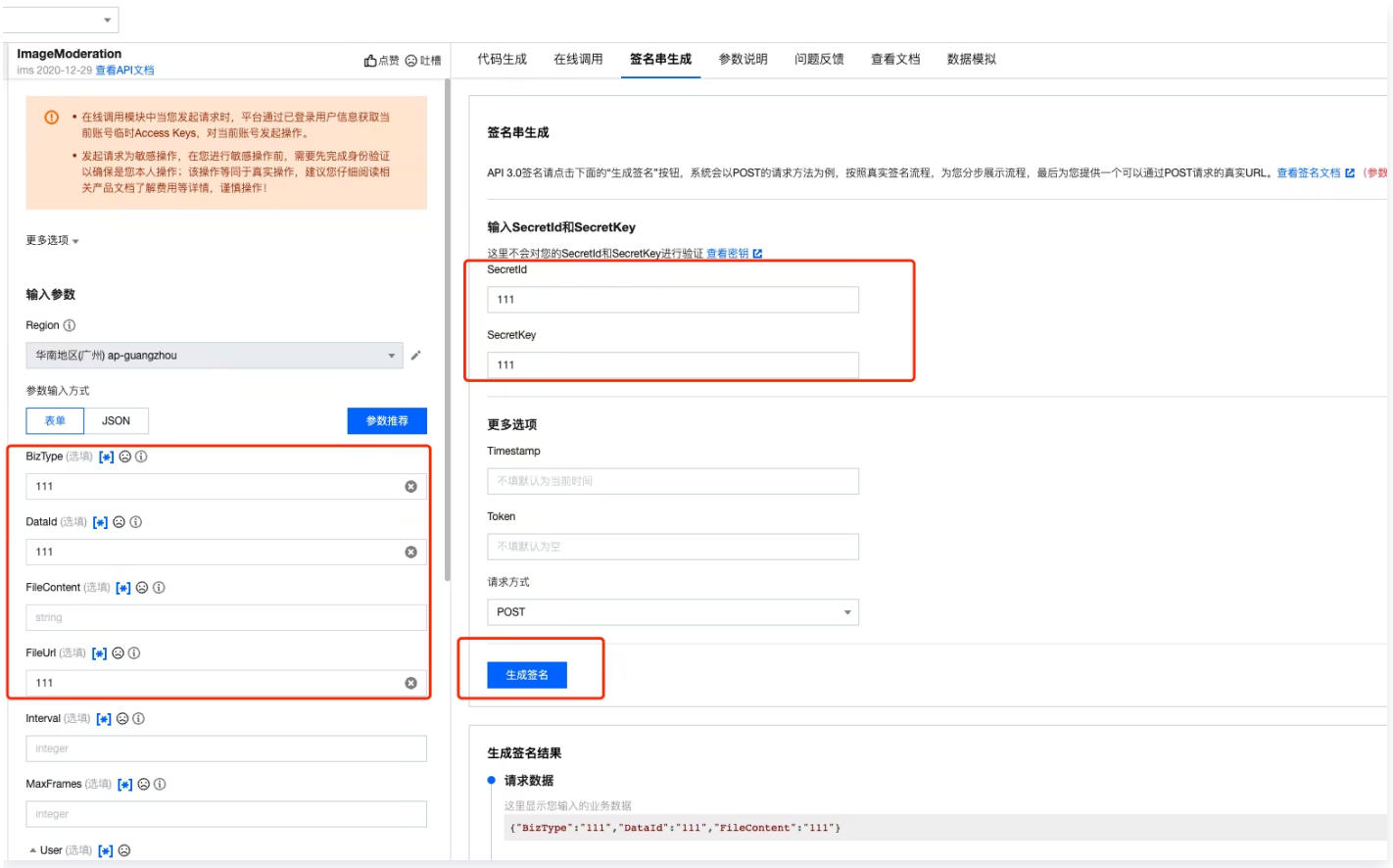
公共参数

公共参数是用于标识用户和接口签名的参数，API 接口文档上不会展示公共参数说明，但每次请求 API 均需要携带这些参数，才能正常发起请求。

签名方法

腾讯云支持 V1、V3两种签名方式，推荐使用签名方法 V3计算签名，签名方法 V3（或被称为 TC3-HMAC-SHA256）相比签名方法 V1更安全，支持更大的请求包，且 POST JSON 格式，性能有一定提升。

首次接触，建议使用 API Explorer 中的“签名串生成”功能，选择签名版本为“API 3.0 签名 V3”，可以生成签名过程进行验证。



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，该字段取值为 ImageModeration。
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。该字段取值为 2020-12-29。
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 ims。 • 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，计算过程请参见 签名版本 v3 签名过程。
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。

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` 协议格式。

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，POST 请求支持 10MB 以内的请求包，签名过程详见 文档 。

示例代码

以下为 Java、Go、Python、PHP 示例 demo，填写密钥信息、接口入参即可调用，其他语言请参考 签名方法 V3-签名演示，签名演示是云服务器服务作为示例，**实际调用时需要根据 API 接口文档填写参数**。

Java

```
import okhttp3.*;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.security.InvalidKeyException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimeZone;

public class ImageModeration {
    public static void main(String[] args) throws IOException,
NoSuchAlgorithmException, InvalidKeyException {
        // 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处
还需注意密钥对的保密
        // 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的
安全性。以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见:
https://cloud.tencent.com/document/product/1278/85305
        // 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi
进行获取
        String secretId = "SecretId";
        String secretKey = "SecretKey";
        String service = "ims";
        String version = "2020-12-29";
```

```
        String action = "ImageModeration";
        String region = "ap-guangzhou";

        // 通过参数传递 BizType 和 FileUrl
        String bizType = "default";
        String fileUrl = "http://example.com/image.jpg"; // 替换为实际的文
件 URL
        String body = createRequestBody(bizType, fileUrl);

        String resp = doRequest(secretId, secretKey, service, version,
action, body, region);
        System.out.println(resp);
    }

    // singleton client for connection reuse and better performance
    private static final OkHttpClient client = new OkHttpClient();

    public static String doRequest(
            String secretId, String secretKey,
            String service, String version, String action,
            String body, String region
    ) throws IOException, NoSuchAlgorithmException, InvalidKeyException
{

        Request request = buildRequest(secretId, secretKey, service,
version, action, body, region);

        System.out.println(request.method() + " " + request.url());
        System.out.println(request.headers());
        System.out.println(body);

        Response response = client.newCall(request).execute();
        return response.body().string();
    }

    public static Request buildRequest(
            String secretId, String secretKey,
            String service, String version, String action,
            String body, String region
    ) throws NoSuchAlgorithmException, InvalidKeyException {
```

```
        String host = "ims.tencentcloudapi.com";
        String endpoint = "https://" + host;
        String contentType = "application/json; charset=utf-8";
        String timestamp = String.valueOf(System.currentTimeMillis() /
1000);
        String auth = getAuth(secretId, secretKey, host, contentType,
timestamp, body);
        return new Request.Builder()
                .header("Host", host)
                .header("X-TC-Timestamp", timestamp)
                .header("X-TC-Version", version)
                .header("X-TC-Action", action)
                .header("X-TC-Region", region)
                .header("X-TC-RequestClient", "SDK_JAVA_BAREBONE")
                .header("Authorization", auth)
                .url(endpoint)
                .post(RequestBody.create(MediaType.parse(contentType),
body))
                .build();
    }

    private static String getAuth(
            String secretId, String secretKey, String host, String
contentType,
            String timestamp, String body
    ) throws NoSuchAlgorithmException, InvalidKeyException {
        String canonicalUri = "/";
        String canonicalQueryString = "";
        String canonicalHeaders = "content-type:" + contentType +
"\nhost:" + host + "\n";
        String signedHeaders = "content-type;host";

        String hashedRequestPayload =
sha256Hex(body.getBytes(StandardCharsets.UTF_8));
        String canonicalRequest = "POST"
                + "\n"
                + canonicalUri
                + "\n"
                + canonicalQueryString
                + "\n"
```

```
                + canonicalHeaders
                + "\n"
                + signedHeaders
                + "\n"
                + hashedRequestPayload;

        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
        String date = sdf.format(new Date(Long.valueOf(timestamp +
"000")));
        String service = host.split("\\.")[0];
        String credentialScope = date + "/" + service + "/" +
"tc3_request";
        String hashedCanonicalRequest =

sha256Hex(canonicalRequest.getBytes(StandardCharsets.UTF_8));
        String stringToSign =
                "TC3-HMAC-SHA256\n" + timestamp + "\n" + credentialScope
+ "\n" + hashedCanonicalRequest;

        byte[] secretDate = hmac256(("TC3" +
secretKey).getBytes(StandardCharsets.UTF_8), date);
        byte[] secretService = hmac256(secretDate, service);
        byte[] secretSigning = hmac256(secretService, "tc3_request");
        String signature =
                printHexBinary(hmac256(secretSigning,
stringToSign)).toLowerCase();
        return "TC3-HMAC-SHA256 "
                + "Credential="
                + secretId
                + "/"
                + credentialScope
                + ", "
                + "SignedHeaders="
                + signedHeaders
                + ", "
                + "Signature="
                + signature;
    }
```

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

    private static final char[] hexCode =
"0123456789ABCDEF".toCharArray();

    public static String printHexBinary(byte[] data) {
        StringBuilder r = new StringBuilder(data.length * 2);
        for (byte b : data) {
            r.append(hexCode[(b >> 4) & 0xF]);
            r.append(hexCode[(b & 0xF)]);
        }
        return r.toString();
    }

    public static byte[] hmac256(byte[] key, String msg) throws
NoSuchAlgorithmException, InvalidKeyException {
        Mac mac = Mac.getInstance("HmacSHA256");
        SecretKeySpec secretKeySpec = new SecretKeySpec(key,
mac.getAlgorithm());
        mac.init(secretKeySpec);
        return mac.doFinal(msg.getBytes(StandardCharsets.UTF_8));
    }

    // 创建请求体的方法
    public static String createRequestBody(String bizType, String
fileUrl) {
        return String.format("{\"BizType\":\"%s\",\"FileUrl\":\"%s\"}",
bizType, fileUrl);
    }
}
```

Go

```go
package ims

import (
    "bytes"
    "crypto/hmac"
    "crypto/sha256"
    "encoding/hex"
    "encoding/json"
    "fmt"
    "log"
    "net/http"
    "strconv"
    "strings"
    "time"
)

// 定义请求参数的结构体
type Payload struct {
    BizType string `json:"BizType"`
    FileUrl string `json:"FileUrl"`
}

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() {
    service := "ims"
    version := "2020-12-29"
    action := "ImageModeration"
    region := "ap-guangzhou"
```

```
    // 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处还需注意密钥对的保密
    // 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全性。以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见：https://cloud.tencent.com/document/product/1278/85305
    // 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi 进行获取
    secretId := "SecretId"
    secretKey := "SecretKey"
    host := "ims.tencentcloudapi.com"
    algorithm := "TC3-HMAC-SHA256"
    var timestamp = time.Now().Unix()

    // ************ 步骤 1：拼接规范请求串 ************
    httpRequestMethod := "POST"
    canonicalURI := "/"
    canonicalQueryString := ""
    contentType := "application/json; charset=utf-8"
    canonicalHeaders := fmt.Sprintf("content-type:%s\nhost:%s\nx-tc-action:%s\n",
        contentType, host, strings.ToLower(action))
    signedHeaders := "content-type;host;x-tc-action"

    // 创建请求参数
    payloadData := Payload{
        BizType: "default",
        FileUrl: "http://example.com/image.jpg", // 这里替换为实际的文件 URL
    }

    // 将payload编码为 JSON
    payloadBytes, err := json.Marshal(payloadData)
    if err != nil {
        log.Fatalf("Error marshaling payload: %v", err)
    }
    payload := string(payloadBytes)

    hashedRequestPayload := sha256hex(payload)
    canonicalRequest := fmt.Sprintf("%s\n%s\n%s\n%s\n%s\n%s",
        httpRequestMethod,
        canonicalURI,
```

```
        canonicalQueryString,
        canonicalHeaders,
        signedHeaders,
        hashedRequestPayload)
    // log.Println(canonicalRequest)

    // ************* 步骤 2：拼接待签名字符串 *************
    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)
    // log.Println(string2sign)

    // ************* 步骤 3：计算签名 *************
    secretDate := hmacsha256(date, "TC3"+secretKey)
    secretService := hmacsha256(service, secretDate)
    secretSigning := hmacsha256("tc3_request", secretService)
    signature := hex.EncodeToString([]byte(hmacsha256(string2sign,
secretSigning)))
    // log.Println(signature)

    // ************* 步骤 4：拼接 Authorization *************
    authorization := fmt.Sprintf("%s Credential=%s/%s, SignedHeaders=%s,
Signature=%s",
        algorithm,
        secretId,
        credentialScope,
        signedHeaders,
        signature)
    // log.Println(authorization)

    // ************* 步骤 5：构造并发起请求 *************
    url := "https://" + host
    httpRequest, _ := http.NewRequest("POST", url,
bytes.NewReader(payloadBytes))
    httpRequest.Header = map[string][]string{
```

```
        "Host":            {host},
        "X-TC-Action":     {action},
        "X-TC-Version":    {version},
        "X-TC-Timestamp": {strconv.FormatInt(timestamp, 10)},
        "Content-Type":    {contentType},
        "Authorization":   {authorization},
    }
    if region != "" {
        httpRequest.Header["X-TC-Region"] = []string{region}
    }
    httpClient := http.Client{}
    resp, err := httpClient.Do(httpRequest)
    if err != nil {
        log.Println(err)
        return
    }
    defer resp.Body.Close()
    body := &bytes.Buffer{}
    _, err = body.ReadFrom(resp.Body)
    if err != nil {
        log.Println(err)
        return
    }
    log.Println(body.String())
}
```

Python

```
#!/usr/bin/env python3
# -*- coding:utf-8 -*-
import hashlib
import hmac
import json
import time
import datetime
import requests
```

```
# 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处还需注意密钥对的保密
# 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全性。以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见:
https://cloud.tencent.com/document/product/1278/85305
# 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi 进行获取
secret_id = "SecretId"
secret_key = "SecretKey"


service = "ims"
host = "ims.tencentcloudapi.com"  # 接口域名
action = "ImageModeration"  # 接口名
endpoint = "https://" + host
region = "ap-guangzhou"  # 地域
version = "2020-12-29"  # 版本为固定值
algorithm = "TC3-HMAC-SHA256"


def do_action(params):
    timestamp = int(time.time())
    day = datetime.datetime.fromtimestamp(timestamp,
datetime.timezone.utc).strftime(
        "%Y-%m-%d"
    )

    # ************* 步骤 1：拼接规范请求串 *************
    http_request_method = "POST"
    canonical_url = "/"
    canonical_querystring = ""
    ct = "application/json; charset=utf-8"
    payload = json.dumps(params)
    canonical_headers = "content-type:%s\nhost:%s\n" % (ct, host)
    signed_headers = "content-type;host"
    hashed_request_payload = hashlib.sha256(payload.encode("utf-
8")).hexdigest()
    canonical_request = (
        http_request_method
        + "\n"
        + canonical_url
```

```
        + "\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 = day + "/" + 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)

    secret_date = sign(("TC3" + secret_key).encode("utf-8"), day)
    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)

    headers = {
        "Authorization": authorization,
        "Content-Type": "application/json; charset=utf-8",
        "Host": host,
        "X-TC-Action": action,
        "X-TC-Timestamp": str(timestamp),
        "X-TC-Version": version,
        "X-TC-Region": region,
    }

    # 发送请求
    resp = requests.post(url=endpoint, data=payload, headers=headers)
    return resp.text


# 计算签名摘要函数
def sign(key, msg):
    return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()


if __name__ == "__main__":
    file_url = "http://example.com/image.jpg" # 这里替换为实际的文件 URL
    params = {"BizType": "default", "FileUrl": file_url}
    resp = do_action(params)
    print(resp)
```

PHP

```php
<?php

function sign($key, $msg) {
    return hash_hmac("sha256", $msg, $key, true);
}

function sendRequest($secret_id, $secret_key, $biz_type, $file_url,
$req_region = "ap-guangzhou") {
    // 定义服务信息
    $service = "ims";
    $host = "ims.tencentcloudapi.com";
    $version = "2020-12-29";
    $action = "ImageModeration";
    $endpoint = "https://ims.tencentcloudapi.com";
    $algorithm = "TC3-HMAC-SHA256";
    $timestamp = time();
    $date = gmdate("Y-m-d", $timestamp);

    // 构造 payload
    $payload_data = [
        "BizType" => $biz_type,
        "FileUrl" => $file_url
    ];
    $payload = json_encode($payload_data);

    // ************* 步骤 1：拼接规范请求串 *************
    $http_request_method = "POST";
    $canonical_uri = "/";
    $canonical_querystring = "";
    $ct = "application/json; charset=utf-8";
    $canonical_headers = "content-type:".$ct."\nhost:".$host."\nx-tc-
action:".strtolower($action)."\n";
    $signed_headers = "content-type;host;x-tc-action";
    $hashed_request_payload = hash("sha256", $payload);
    $canonical_request =
"$http_request_method\n$canonical_uri\n$canonical_querystring\n$canonica
l_headers\n$signed_headers\n$hashed_request_payload";
```

```
    // ************* 步骤 2：拼接待签名字符串 *************
    $credential_scope = "$date/$service/tc3_request";
    $hashed_canonical_request = hash("sha256", $canonical_request);
    $string_to_sign =
"$algorithm\n$timestamp\n$credential_scope\n$hashed_canonical_request";

    // ************* 步骤 3：计算签名 *************
    $secret_date = sign("TC3".$secret_key, $date);
    $secret_service = sign($secret_date, $service);
    $secret_signing = sign($secret_service, "tc3_request");
    $signature = hash_hmac("sha256", $string_to_sign, $secret_signing);

    // ************* 步骤 4：拼接 Authorization *************
    $authorization = "$algorithm
Credential=$secret_id/$credential_scope, SignedHeaders=$signed_headers,
Signature=$signature";

    // ************* 步骤 5：构造并发起请求 *************
    $headers = [
        "Authorization" => $authorization,
        "Content-Type" => "application/json; charset=utf-8",
        "Host" => $host,
        "X-TC-Action" => $action,
        "X-TC-Timestamp" => $timestamp,
        "X-TC-Version" => $version
    ];
    if ($req_region) {
        $headers["X-TC-Region"] = $req_region;
    }

    try {
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL, $endpoint);
        curl_setopt($ch, CURLOPT_POST, true);
        curl_setopt($ch, CURLOPT_POSTFIELDS, $payload);
        curl_setopt($ch, CURLOPT_HTTPHEADER, array_map(function ($k, $v)
{ return "$k: $v"; }, array_keys($headers), $headers));
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
        $response = curl_exec($ch);
```

```
        if ($response === FALSE) {
            throw new Exception(curl_error($ch), curl_errno($ch));
        }
        curl_close($ch);

        return $response;
    } catch (Exception $err) {
        echo 'Curl error: ' . $err->getMessage();
    }
}

// 使用示例
$biz_type = "default";
$file_url = "http://example.com/image.jpg"; // 这里替换为实际的文件 URL

try {
    // 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处还需注意密钥对的保密
    // 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全性。以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见：https://cloud.tencent.com/document/product/1278/85305
    // 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi 进行获取
    $response = sendRequest("SecretId", "SecretKey", $biz_type, $file_url);
    echo $response;
} catch (Exception $err) {
    echo $err->getMessage();
}

?>
```

签名失败

存在以下签名失败的错误码，请根据实际情况处理。

错误码	错误描述
AuthFailure.SignatureExpire	签名过期。Timestamp 与服务器接收到请求的时间相差不得超过五分钟。

AuthFailure.SecretIdNotFound	密钥不存在。请到 控制台 查看密钥是否被禁用，是否少复制了字符或者多了字符。
AuthFailure.SignatureFailure	签名错误。可能是签名计算错误，或者签名与实际发送的内容不相符合，也有可能是密钥 SecretKey 错误导致的。
AuthFailure.TokenFailure	临时证书 Token 错误。
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）。

SDK 接入指引

最近更新时间：2026-09-08 17:18:33

推荐使用腾讯云 API 配套的 7 种常见的编程语言 SDK，已经封装了签名和请求过程，开发时只需关注产品提供的具体接口即可。

API 文档

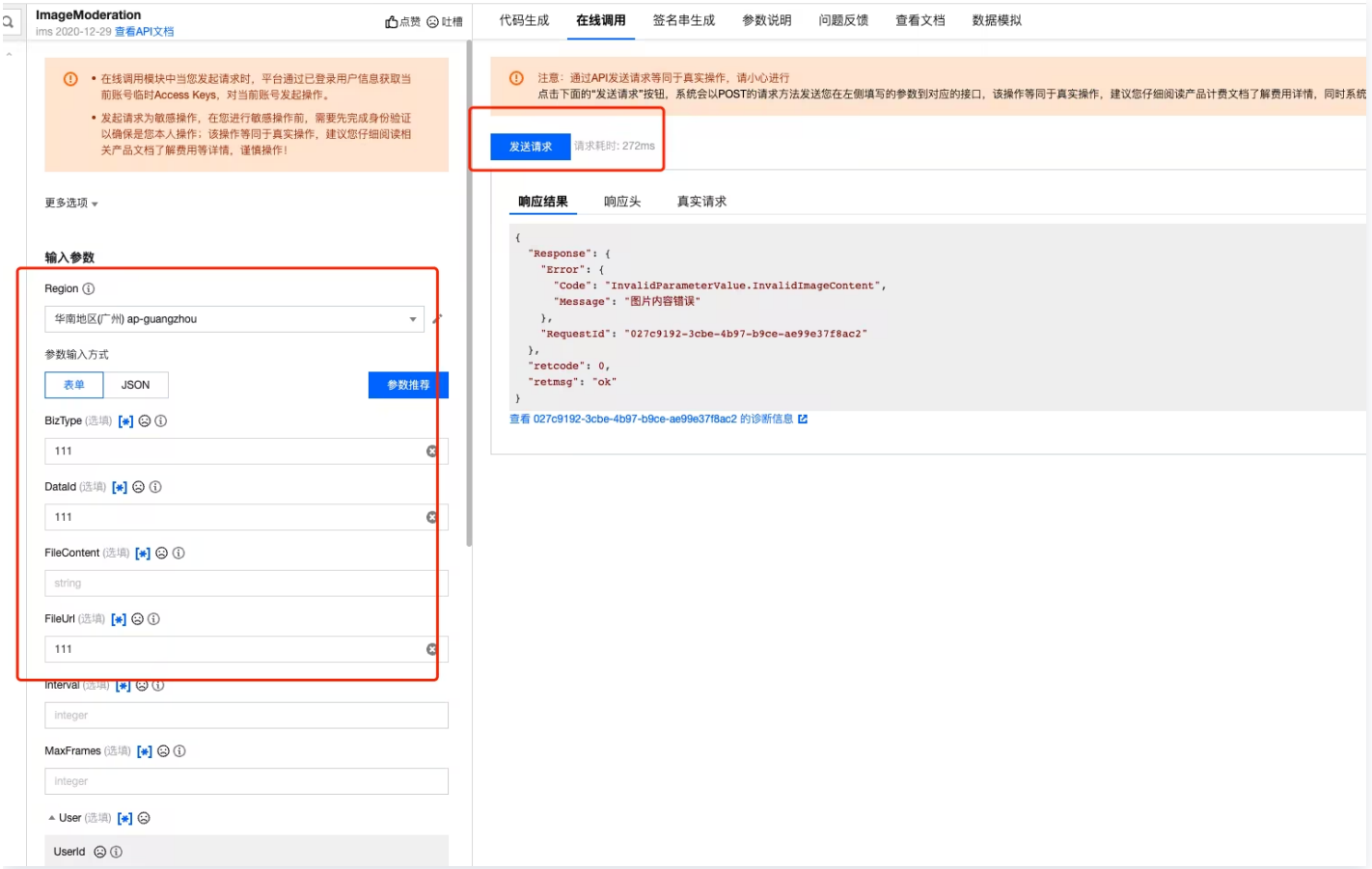
参考 图片内容安全 接口文档，了解需要使用的参数。

SDK 使用说明

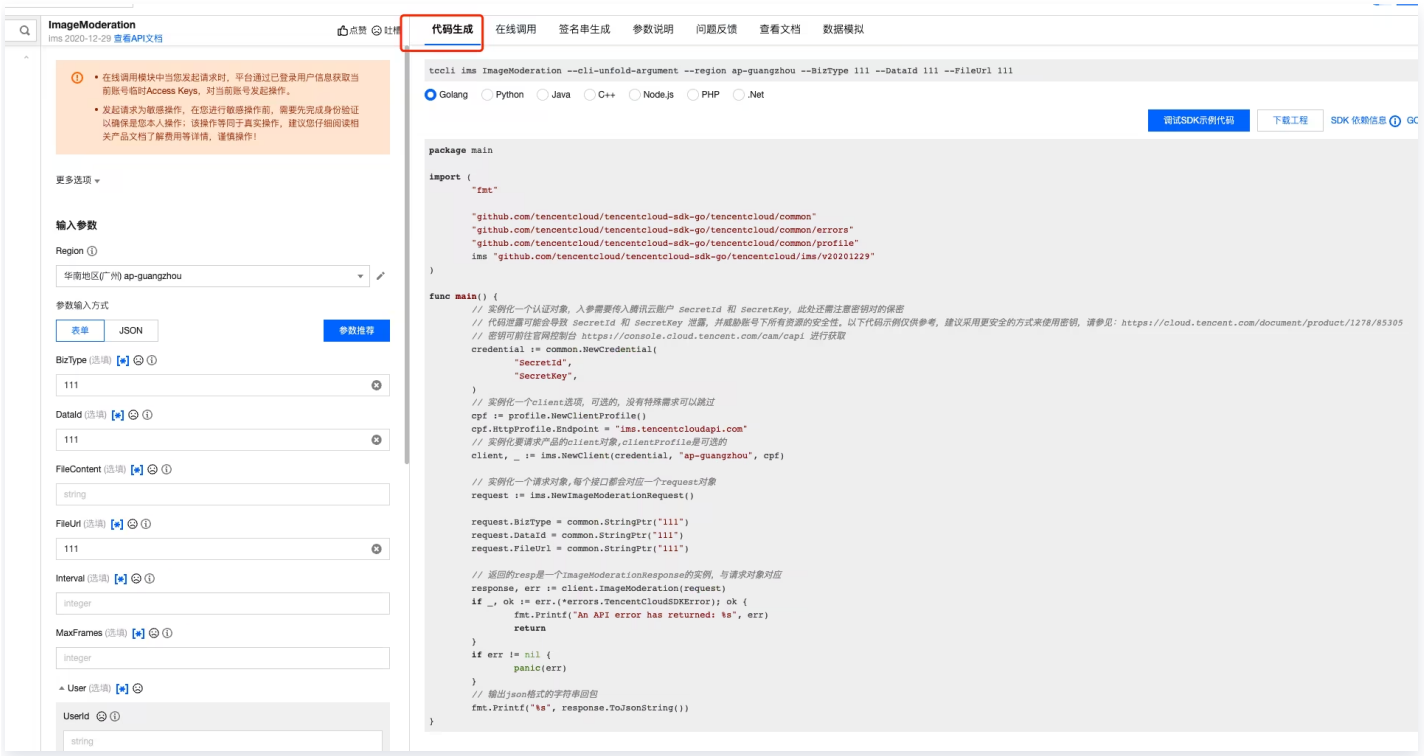
云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，均已开源，能更方便的调用 API，支持 Python 、Java 、PHP 、Go 、NodeJS 、.NET 、C++ 。
各语言 SDK 使用说明可参考 文档，介绍依赖版本以及安装过程等。

建议安装最新版本的 SDK，即可调用接口，腾讯云提供了 API Explorer 可在线调用，并且可以快速生成 SDK 调用示例代码，以下 Java、Python、Golang 示例均从 API Explorer 生成，其他语言可自行调试获取。

在线调用示例



代码生成示例



Java

```
import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.profile.ClientProfile;
import com.tencentcloudapi.common.profile.HttpProfile;
import com.tencentcloudapi.common.exception.TencentCloudSDKException;
import com.tencentcloudapi.ims.v20201229.ImsClient;
import com.tencentcloudapi.ims.v20201229.models.*;

public class ImageModeration
{
    public static void main(String [] args) {
        try{
            // 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处还需注意密钥对的保密
            // 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全性。以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见：https://cloud.tencent.com/document/product/1278/85305
            // 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi 进行获取
```

```
            Credential cred = new Credential("SecretId", "SecretKey");
            // 实例化一个http选项，可选的，没有特殊需求可以跳过
            HttpProfile httpProfile = new HttpProfile();
            httpProfile.setEndpoint("ims.tencentcloudapi.com");
            // 实例化一个client选项，可选的，没有特殊需求可以跳过
            ClientProfile clientProfile = new ClientProfile();
            clientProfile.setHttpProfile(httpProfile);
            // 实例化要请求产品的client对象,clientProfile是可选的
            ImsClient client = new ImsClient(cred, "ap-guangzhou",
clientProfile);
            // 实例化一个请求对象,每个接口都会对应一个request对象
            ImageModerationRequest req = new ImageModerationRequest();
            req.setBizType("default");

req.setFileUrl("https://xxxxx.com.cn/EQjp8HObL2VLbxEx/fc/open/image/2022
092921/xxx.jpg");
            // 返回的resp是一个ImageModerationResponse的实例，与请求对象对应
            ImageModerationResponse resp = client.ImageModeration(req);
            // 输出json格式的字符串回包

System.out.println(ImageModerationResponse.toJsonString(resp));
        } catch (TencentCloudSDKException e) {
            System.out.println(e.toString());
        }
    }
}
```

Python

```
import json
from tencentcloud.common import credential
from tencentcloud.common.profile.client_profile import ClientProfile
from tencentcloud.common.profile.http_profile import HttpProfile
from tencentcloud.common.exception.tencent_cloud_sdk_exception import
TencentCloudSDKException
from tencentcloud.ims.v20201229 import ims_client, models
try:
    # 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处还需注
意密钥对的保密
```

```
    # 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全
性。以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见:
https://cloud.tencent.com/document/product/1278/85305
    # 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi 进行获
取
    cred = credential.Credential("SecretId", "SecretKey")
    # 实例化一个http选项，可选的，没有特殊需求可以跳过
    httpProfile = HttpProfile()
    httpProfile.endpoint = "ims.tencentcloudapi.com"

    # 实例化一个client选项，可选的，没有特殊需求可以跳过
    clientProfile = ClientProfile()
    clientProfile.httpProfile = httpProfile
    # 实例化要请求产品的client对象,clientProfile是可选的
    client = ims_client.ImsClient(cred, "ap-guangzhou", clientProfile)

    # 实例化一个请求对象,每个接口都会对应一个request对象
    req = models.ImageModerationRequest()
    params = {
        "BizType": "default",
        "FileUrl":
"https://xxxxx.com.cn/EQjp8HObL2VLbxEx/fc/open/image/2022092921/xxx.jpg"
    }
    req.from_json_string(json.dumps(params))

    # 返回的resp是一个ImageModerationResponse的实例，与请求对象对应
    resp = client.ImageModeration(req)
    # 输出json格式的字符串回包
    print(resp.to_json_string())

except TencentCloudSDKException as err:
    print(err)
```

Go

```
package main

import (
        "fmt"
```

```
        "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common"
        "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/errors"
        "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/profile"
        ims "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/ims/v20201229"
)

func main() {
        // 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处
还需注意密钥对的保密
        // 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的
安全性。以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见:
https://cloud.tencent.com/document/product/1278/85305
        // 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi
进行获取
        credential := common.NewCredential(
                "SecretId",
                "SecretKey",
        )
        // 实例化一个client选项，可选的，没有特殊需求可以跳过
        cpf := profile.NewClientProfile()
        cpf.HttpProfile.Endpoint = "ims.tencentcloudapi.com"
        // 实例化要请求产品的client对象,clientProfile是可选的
        client, _ := ims.NewClient(credential, "ap-guangzhou", cpf)

        // 实例化一个请求对象,每个接口都会对应一个request对象
        request := ims.NewImageModerationRequest()

        request.BizType = common.StringPtr("default")
        request.FileUrl =
common.StringPtr("https://xxxxx.com.cn/EQjp8HObL2VLbxEx/fc/open/image/20
22092921/xxx.jpg")

        // 返回的resp是一个ImageModerationResponse的实例，与请求对象对应
        response, err := client.ImageModeration(request)
        if _, ok := err.(*errors.TencentCloudSDKError); ok {
```

```
                fmt.Printf("An API error has returned: %s", err)
                return
        }
        if err != nil {
                panic(err)
        }
        // 输出json格式的字符串回包
        fmt.Printf("%s", response.ToJsonString())
}
```

热点问题

SDK 是否支持设置超时时间?

SDK 具有默认的超时时间，除非必要，请勿更改默认设置。如果需要修改，可以参考 SDK 仓库 中的详细示例。

详细版

```java
import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.exception.TencentCloudSDKException;
// 导入对应产品模块的client
import com.tencentcloudapi.cvm.v20170312.CvmClient;
// 导入要请求接口对应的request response类
import com.tencentcloudapi.cvm.v20170312.models.DescribeInstancesRequest;
import com.tencentcloudapi.cvm.v20170312.models.DescribeInstancesResponse;
import com.tencentcloudapi.cvm.v20170312.models.Filter;
//导入可选配置类
import com.tencentcloudapi.common.profile.ClientProfile;
import com.tencentcloudapi.common.profile.HttpProfile;
import com.tencentcloudapi.common.profile.Language;

public class DescribeInstances {
    public static void main(String[] args) {
        try {
            // 实例化一个认证对象，入参需要传入腾讯云账户 SecretId，SecretKey。
            // 为了保护密钥安全，建议将密钥设置在环境变量中或者配置文件中，请参考本文凭证管理章节。
            // 硬编码密钥到代码中有可能随代码泄露而暴露，有安全隐患，并不推荐。
            // Credential cred = new Credential("SecretId", "SecretKey");
            Credential cred = new Credential(System.getenv("TENCENTCLOUD_SECRET_ID"), System.getenv(

            // 实例化一个http选项，可选的，没有特殊需求可以跳过
            HttpProfile httpProfile = new HttpProfile();
            // 从3.0.96版本开始，单独设置 HTTP 代理
            // httpProfile.setProxyHost("真实代理ip");
            // httpProfile.setProxyPort(真实代理端口);
            httpProfile.setReqMethod("GET"); // get请求(默认为post请求)
            httpProfile.setConnTimeout(30); // 请求连接超时时间，单位为秒(默认60秒)
            httpProfile.setWriteTimeout(30);  // 设置写入超时时间，单位为秒(默认0秒)
            httpProfile.setReadTimeout(30);  // 设置读取超时时间，单位为秒(默认0秒)
            httpProfile.setEndpoint("cvm.ap-shanghai.tencentcloudapi.com"); // 指定接入地域域名(默认就近
```

多线程是否可以共用一个 Client?

SDK 目前不支持线程安全，因此需要在每个线程中创建一个 Client 来进行请求。

同时连接数量是否有限制?

目前没有限制，但是接口会限制每秒请求数（QPS）的并发数，文本接口默认为1000QPS。建议客户端也控制并发数，因为太多线程可能超出用户机器的处理能力，导致连接失败。

接入后验证

最近更新时间：2026-09-08 17:18:33

当您完成 API 或 SDK 接入后，可以通过如下方式验证图片内容检测服务是否接入成功。

接入成功

若 Response 回参没有“Error”字段且正常返回业务参数，则说明 IMS 接口接入成功，示例如下所示：


```
{
  "Response": {
    "RequestId": "a61237dd-c2a0-43e7-a3da-d27022d39ba7",
    "DataId": "a61237dd-c2a0-43e7-a3da-d27022d39ba7",
    "BizType": "test_1001",
    "Suggestion": "Block",
    "FileMD5": "",
    "Label": "Porn",
    "SubLabel": "SexBehavior",
    "Score": 90,
    "LabelResults": [
      {
        "Scene": "Porn",
        "Suggestion": "Block",
        "Label": "Porn",
        "SubLabel": "SexBehavior",
        "Score": 90,
        "Details": []
      }
    ],
    "ObjectResults": [
      {
        "Scene": "QrCode",
        "Suggestion": "Block",
        "Label": "Ad",
        "SubLabel": "",
        "Score": 100,
        "Names": [
          "QRCODE"
        ],
```

```
        "Details": [
          {
            "Id": 0,
            "Name": "QRCODE",
            "Score": 100,
            "Location": {
              "X": 155.01746,
              "Y": 396.01746,
              "Width": 769.9824,
              "Height": 769.98254,
              "Rotate": 0
            }
          }
        ]
      }
    ],
    "OcrResults": [],
    "LibResults": [],
    "Extra": ""
  }
}
```


接入失败

若 Response 回参显示“Error”字段，则说明接入失败，以下为 signature 校验失败的示例：


```
{
    "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"
    }
}
```

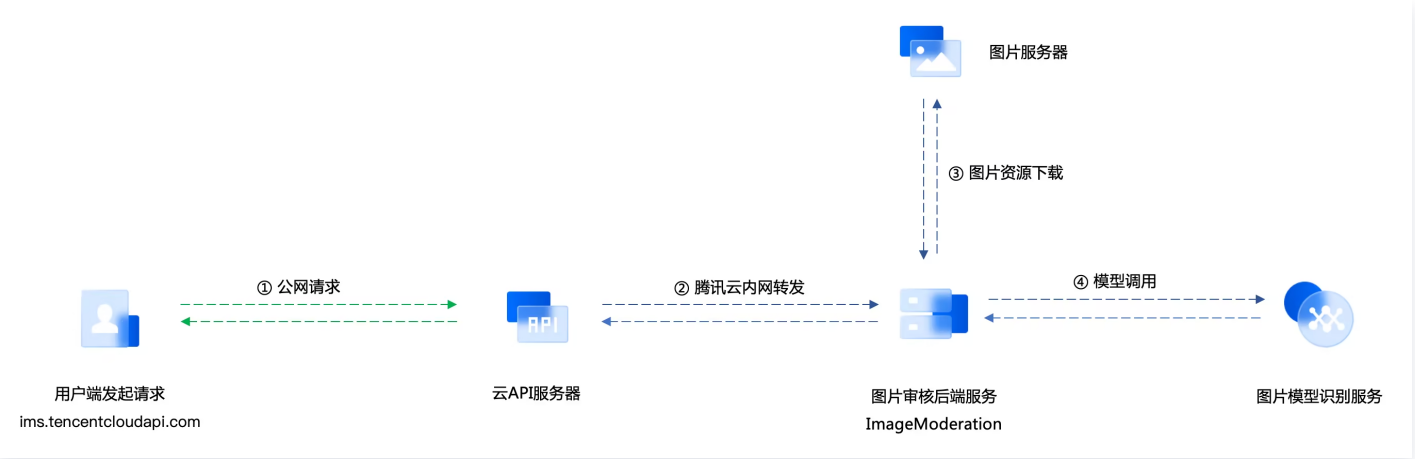

其中 Error 中的 Code 表示错误码，Message 表示错误的具体信息，详情请参见 错误码 。

耗时说明

最近更新时间：2026-09-08 17:18:33

本文档将介绍图片内容安全（Image Moderation System，IMS）进行图片审核过程中，各阶段耗时的简要说明。

图片审核请求路径



图片审核耗时说明

整体审核耗时

①公网请求：用户端向图片接口域名 `ims.tencentcloudapi.com` 发起请求，经由公网请求到达云 API 服务器阶段的耗时，此阶段耗时可能受公网网络质量、CDN 网络波动等网络情况影响。
②腾讯云内网转发：请求到达云 API 服务器，由腾讯云内网转发至图片审核后端服务阶段的耗时。
③图片资源下载：图片审核后端服务访问 FileUrl 送审图片链接进行图片下载阶段的耗时，下载耗时可能受图片文件大小、公网网络质量、CDN 网络波动等因素影响。

> **说明：**
> 若采用上传图片文件进行送审，则可以省略此③图片资源下载步骤，图片文件大小支持：5MB 以内。

④模型调用：图片审核后端服务调用图片模型识别服务进行图片审核阶段的耗时。

耗时说明

字段含义为：耗时说明字段包括②、③、④阶段的总耗时，即云 API 从收到用户端请求开始，至审核完成的结果从云 API 发出结束，期间花费的耗时。

音频内容安全接入指引
API接入指引

最近更新时间：2026-09-08 17:18:33

腾讯云 API 会对每个请求进行身份验证，用户使用原生 API 接入需要使用安全凭证，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，以下内容主要介绍 API 接口文档参数以及 V3签名方法。

API 文档

参考 音频内容安全 接口文档，了解请求 API 所需要的公共参数、域名、接口输入输出参数值。

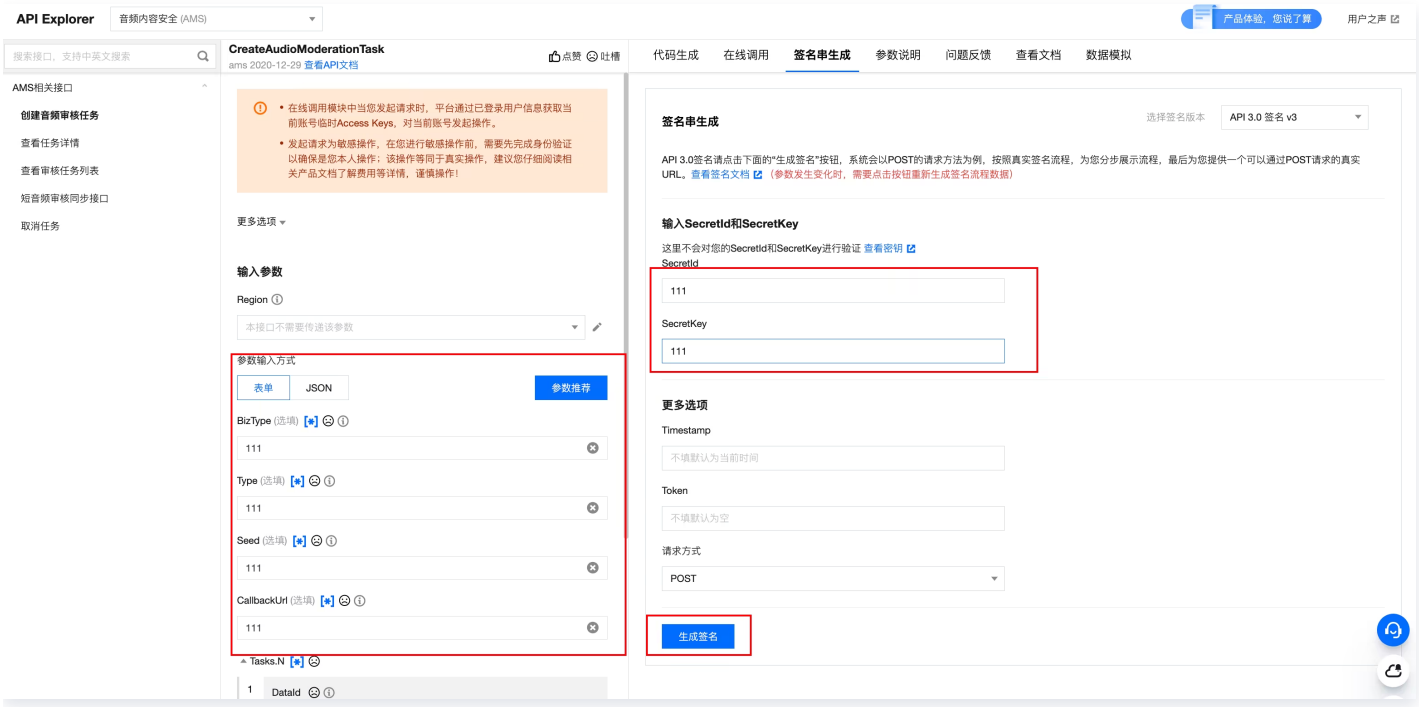
公共参数

公共参数是用于标识用户和接口签名的参数，API 接口文档上不会展示公共参数说明，但每次请求 API 均需要携带这些参数，才能正常发起请求。

签名方法

腾讯云支持 V1、V3两种签名方式，推荐使用签名方法 V3计算签名，签名方法 V3（或被称为 TC3-HMAC-SHA256）相比签名方法 V1更安全，支持更大的请求包，且 POST JSON 格式，性能有一定提升。

首次接触，建议使用 API Explorer 中的“签名串生成”功能，选择签名版本为“API 3.0 签名 V3”，可以生成签名过程进行验证。



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，该字段取值为 CreateAudioModerationTask。
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。该字段取值为 2020-12-29。
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 ams。 • 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，计算过程请参见 签名版本 v3 签名过程 。
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。

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` 协议格式。

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，POST 请求支持 10MB 以内的请求包，签名过程详见 文档 。

示例代码

以下为 Java、Go、Python 示例 demo，填写密钥信息、接口入参即可调用，其他语言请参考 签名方法 V3-签名演示 ，签名演示是云服务器服务作为示例，**实际调用时需要根据 API 接口文档填写参数**。

Java

```
import java.nio.charset.Charset;
import java.io.OutputStream;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
```

```
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 CreateAudioModeration {
    private final static Charset UTF8 = StandardCharsets.UTF_8;
    private final static String SECRET_ID = "";
    private final static String 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 = "ams";
        String host = "ams.tencentcloudapi.com";
        String region = "ap-guangzhou";
        String action = "CreateAudioModeration";
        String version = "2020-12-29";
        String algorithm = "TC3-HMAC-SHA256";
```

```
        String timestamp = String.valueOf(System.currentTimeMillis() /
1000);
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
        String date = sdf.format(new Date(Long.valueOf(timestamp +
"000")));

        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 = "{\"Type\": \"xxxx\",\"BizType\": \"xxxx\"}";//
body数据为json
        String hashedRequestPayload = sha256Hex(payload);
        String canonicalRequest = httpRequestMethod + "\n" +
canonicalUri + "\n" + canonicalQueryString + "\n"
                + canonicalHeaders + "\n" + signedHeaders + "\n" +
hashedRequestPayload;

        String credentialScope = date + "/" + service + "/" +
"tc3_request";
        String hashedCanonicalRequest = sha256Hex(canonicalRequest);
        String stringToSign = algorithm + "\n" + timestamp + "\n" +
credentialScope + "\n" + hashedCanonicalRequest;

        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();

        String authorization = algorithm + " " + "Credential=" +
SECRET_ID + "/" + credentialScope + ", "
```

```
                + "SignedHeaders=" + signedHeaders + ", " + "Signature="
+ signature;

        try {
            URL url = new URL("https://" + host);
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            connection.setRequestMethod("POST");
            connection.setRequestProperty("Authorization",
authorization);
            connection.setRequestProperty("Content-Type", CT_JSON);
            connection.setRequestProperty("Host", host);
            connection.setRequestProperty("X-TC-Action", action);
            connection.setRequestProperty("X-TC-Timestamp", timestamp);
            connection.setRequestProperty("X-TC-Version", version);
            connection.setRequestProperty("X-TC-Region", region);

            // Enable input/output streams and write the payload
            connection.setDoOutput(true);
            OutputStream os = connection.getOutputStream();
            os.write(payload.getBytes(UTF8));
            os.flush();
            os.close();

            // Get the response
            int responseCode = connection.getResponseCode();
            if (responseCode == 200) {
                BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
                String inputLine;
                StringBuilder response = new StringBuilder();

                while ((inputLine = in.readLine()) != null) {
                    response.append(inputLine);
                }
                in.close();

                // Process the response as needed
                System.out.println("Response: " + response.toString());
            } else {
```

```
                // Handle the error response
                System.out.println("Error Response Code: " +
responseCode);
            }

            connection.disconnect();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Go

```
package ams

import (
    "crypto/hmac"
    "crypto/sha256"
    "encoding/hex"
    "encoding/json"
    "fmt"
    "io/ioutil"
    "net/http"
    "strconv"
    "strings"
    "time"
)

type createmod struct {
    BizType string `json:"BizType"`
    Type    string `json:"Type"`
    Tasks   []task
}
type task struct {
    DataId string    `json:"DataId"`
    Input  taskInput `json:"Input"`
}
```

```
type taskInput struct {
    Type string `json:"Type"`
    Url  string `json:"Url"`
}

func sha256heCreate(s string) string {
    b := sha256.Sum256([]byte(s))
    return hex.EncodeToString(b[:])
}

func hmacsha256Create(s, key string) string {
    hashed := hmac.New(sha256.New, []byte(key))
    hashed.Write([]byte(s))
    return string(hashed.Sum(nil))
}

func DetectCreateSingle() {
    secretId := ""
    secretKey := ""
    host := "ams.tencentcloudapi.com"
    algorithm := "TC3-HMAC-SHA256"
    service := "ams"
    version := "2020-12-29"
    action := "CreateAudioModerationTask"
    region := "ap-guangzhou"
    var timestamp int64 = time.Now().Unix()
    // loc, _ := time.LoadLocation("Asia/Shanghai")
    // var timestamp int64 = time.Now().In(loc).Unix()

    // step 1: build canonical request string
    httpRequestMethod := "POST"
    canonicalURI := "/"
    canonicalQueryString := ""
    canonicalHeaders := "content-type:application/json; charset=utf-8\n"
+ "host:" + host + "\n"
    signedHeaders := "content-type;host"
    var a = createmod{
        BizType: "default",
        Type:    "AUDIO",
        Tasks: []task{
```

```
            {
                DataId: "test",
                Input: taskInput{
                    Type: "URL",
                    Url:  "", // 接口入参
                },
            },
        },
    }
    b, err := json.Marshal(a)
    if err != nil {
        fmt.Print(err.Error())
        return
    }
    payload := string(b)
    hashedRequestPayload := sha256heCreate(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 := sha256heCreate(canonicalRequest)
    string2sign := fmt.Sprintf("%s\n%d\n%s\n%s",
        algorithm,
        timestamp,
        credentialScope,
        hashedCanonicalRequest)
    // fmt.Println(string2sign)

    // step 3: sign string
    secretDate := hmacsha256Create(date, "TC3"+secretKey)
    secretService := hmacsha256Create(service, secretDate)
    secretSigning := hmacsha256Create("tc3_request", secretService)
```

```
    signature := hex.EncodeToString([]byte(hmacsha256Create(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)

    req, _ := http.NewRequest("POST", "https://"+host,
strings.NewReader(payload))
    req.Header.Set("Authorization", authorization)
    req.Header.Set("Content-Type", "application/json; charset=utf-8")
    req.Header.Set("Host", host)
    req.Header.Set("X-TC-Action", action)
    req.Header.Set("X-TC-Timestamp", strconv.FormatInt(timestamp, 10))
    req.Header.Set("X-TC-Version", version)
    req.Header.Set("X-TC-Region", region)

    resp, err := (&http.Client{}).Do(req)
    if err != nil {
        fmt.Print(err.Error())
        return
    }
    defer resp.Body.Close()

    respByte, _ := ioutil.ReadAll(resp.Body)
    fmt.Println(string(respByte))
}
```

Python

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import hashlib
```

```
import hmac
import json
import time
from datetime import datetime
import requests
from concurrent.futures import ThreadPoolExecutor

secret_id = ""
secret_key = ""

service = "ams"
host = "ams.tencentcloudapi.com"
endpoint = "https://" + host
region = "ap-guangzhou"
version = "2020-12-29"
algorithm = "TC3-HMAC-SHA256"


def do_action(action, params):
    timestamp = int(time.time())
    day = datetime.utcfromtimestamp(timestamp).strftime("%Y-%m-%d")

    # ************* 步骤 1：拼接规范请求串 *************
    http_request_method = "POST"
    canonical_uri = "/"
    canonical_querystring = ""
    ct = "application/json; charset=utf-8"
    payload = json.dumps(params)
    canonical_headers = "content-type:%s\nhost:%s\n" % (ct, host)
    signed_headers = "content-type;host"
    hashed_request_payload = hashlib.sha256(
        payload.encode("utf-8")).hexdigest()
    canonical_request = (http_request_method + "\n" + canonical_uri +
"\n" +
                         canonical_querystring + "\n" +
canonical_headers +
                         "\n" + signed_headers + "\n" +
hashed_request_payload)
    # print(canonical_request)
```

```
    # ************* 步骤 2：拼接待签名字符串 *************
    credential_scope = day + "/" + 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)

    secret_date = sign(("TC3" + secret_key).encode("utf-8"), day)
    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)

    headers = {
        "Authorization": authorization,
        "Content-Type": "application/json; charset=utf-8",
        "Host": host,
        "X-TC-Action": action,
        "X-TC-Timestamp": str(timestamp),
        "X-TC-Version": version,
        "X-TC-Region": region
    }

    # 发送请求
    resp = requests.post(url=endpoint, data=payload, headers=headers)

    return resp


# 计算签名摘要函数
```

```
def sign(key, msg):
    return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()


def audio_detect(params):
    action = "CreateAudioModerationTask"
    resp = do_action(action, params)

    return resp.text


if __name__ == "__main__":
    params = []
    audio_url = '' # 接口参数
    p = {
        "BizType": "default",
        'Type': 'AUDIO',
        'Tasks': [{
            "DataId": "case",
            'Input': {
                'Type': 'URL',
                'Url': audio_url
            }
        }]
    }
    params.append(p)

    executor = ThreadPoolExecutor(max_workers=5)
    resps = executor.map(audio_detect, params)
    for resp in resps:
        print(resp)
```

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

SDK 接入指引

最近更新时间：2026-09-08 17:18:32

推荐使用腾讯云 API 配套的 7 种常见的编程语言 SDK，已经封装了签名和请求过程，开发时只关注产品提供的具体接口即可。

API 文档

参考 音频内容安全 接口文档，了解需要使用的参数。

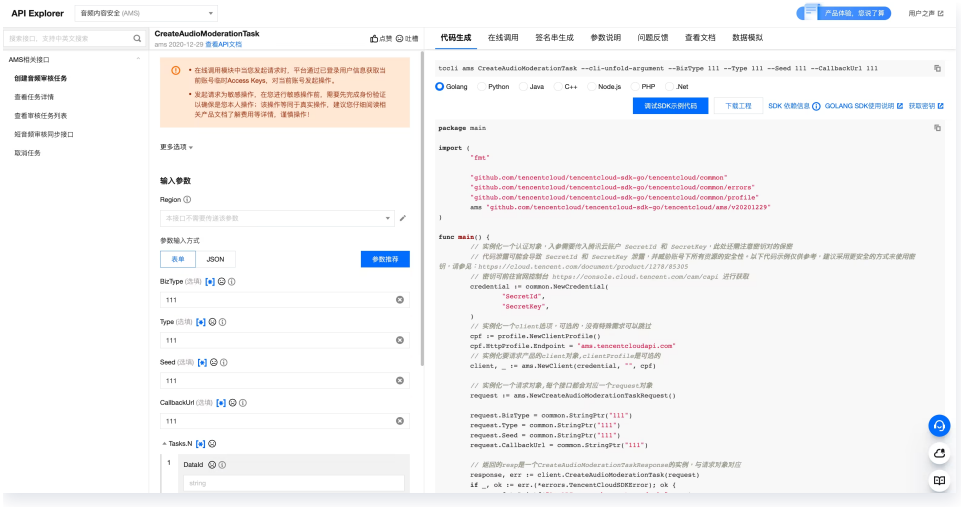
SDK 使用说明

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，均已开源，能更方便的调用 API，支持 Python 、Java 、PHP 、Go 、NodeJS 、.NET 、C++ 。
各语言 SDK 使用说明可参考 文档，介绍依赖版本以及安装过程等。

建议安装最新版本的 SDK，即可调用接口，腾讯云提供了 API Explorer 可在线调用，并且可以快速生成 SDK 调用示例代码，以下 Java、python、Golang 示例均从 API Explorer 生成，其他语言可自行调试获取。

在线调用示例

代码生成示例



Java

```
import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.profile.ClientProfile;
import com.tencentcloudapi.common.profile.HttpProfile;
import com.tencentcloudapi.common.exception.TencentCloudSDKException;
import com.tencentcloudapi.vm.v20210922.VmClient;
import com.tencentcloudapi.vm.v20210922.models.*;

public class CreateVideoModerationTask
{
    public static void main(String [] args) {
        try{
            // 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处还需注意密钥对的保密
            // 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全性。以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见：https://cloud.tencent.com/document/product/1278/85305
            // 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi 进行获取
            Credential cred = new Credential("SecretId", "SecretKey");
            // 实例化一个http选项，可选的，没有特殊需求可以跳过
            HttpProfile httpProfile = new HttpProfile();
            httpProfile.setEndpoint("vm.tencentcloudapi.com");
            // 实例化一个client选项，可选的，没有特殊需求可以跳过
            ClientProfile clientProfile = new ClientProfile();
            clientProfile.setHttpProfile(httpProfile);
            // 实例化要请求产品的client对象,clientProfile是可选的
```

```
            VmClient client = new VmClient(cred, "ap-guangzhou",
clientProfile);
            // 实例化一个请求对象,每个接口都会对应一个request对象
            CreateVideoModerationTaskRequest req = new
CreateVideoModerationTaskRequest();
            req.setBizType("default");
            req.setType("VIDEO");

            TaskInput[] taskInputs1 = new TaskInput[1];
            TaskInput taskInput1 = new TaskInput();
            taskInput1.setDataId("test");
            taskInput1.setName("test");
            StorageInfo storageInfo1 = new StorageInfo();
            storageInfo1.setType("URL");
            storageInfo1.setUrl("http://xxx.mp4");
            taskInput1.setInput(storageInfo1);

            taskInputs1[0] = taskInput1;

            req.setTasks(taskInputs1);

            // 返回的resp是一个CreateVideoModerationTaskResponse的实例，与请
求对象对应
            CreateVideoModerationTaskResponse resp =
client.CreateVideoModerationTask(req);
            // 输出json格式的字符串回包

System.out.println(CreateVideoModerationTaskResponse.toJsonString(resp))
;
        } catch (TencentCloudSDKException e) {
            System.out.println(e.toString());
        }
    }
}
```

Python

```
import json
from tencentcloud.common import credential
```

```
from tencentcloud.common.profile.client_profile import ClientProfile
from tencentcloud.common.profile.http_profile import HttpProfile
from tencentcloud.common.exception.tencent_cloud_sdk_exception import
TencentCloudSDKException
from tencentcloud.ams.v20201229 import ams_client, models
try:
    # 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处还需注
意密钥对的保密
    # 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全
性。以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见:
https://cloud.tencent.com/document/product/1278/85305
    # 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi 进行获
取
    cred = credential.Credential("SecretId", "SecretKey")
    # 实例化一个http选项，可选的，没有特殊需求可以跳过
    httpProfile = HttpProfile()
    httpProfile.endpoint = "ams.tencentcloudapi.com"

    # 实例化一个client选项，可选的，没有特殊需求可以跳过
    clientProfile = ClientProfile()
    clientProfile.httpProfile = httpProfile
    # 实例化要请求产品的client对象,clientProfile是可选的
    client = ams_client.AmsClient(cred, "", clientProfile)

    # 实例化一个请求对象,每个接口都会对应一个request对象
    req = models.CreateAudioModerationTaskRequest()
    params = {
        "BizType": "default",
        "Type": "AUDIO",
        "Tasks": [
            {
                "DataId": "111",
                "Name": "111",
                "Input": {
                    "Type": "URL",
                    "Url":
"https://xxxxx.com.cn/EQjp8HObL2VLbxEx/fc/open/image/2022092921/xxx.jpg"
                }
            }
        ]
```

```
    }
    req.from_json_string(json.dumps(params))

    # 返回的resp是一个CreateAudioModerationTaskResponse的实例，与请求对象对应
    resp = client.CreateAudioModerationTask(req)
    # 输出json格式的字符串回包
    print(resp.to_json_string())

except TencentCloudSDKException as err:
    print(err)
```

Go

```
package main

import (
        "fmt"

        "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common"
        "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/errors"
        "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/profile"
        ams "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/ams/v20201229"
)

func main() {
        // 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处
还需注意密钥对的保密
        // 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的
安全性。以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见:
https://cloud.tencent.com/document/product/1278/85305
        // 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi
进行获取
        credential := common.NewCredential(
                "SecretId",
                "SecretKey"
```

```
        )
        // 实例化一个client选项，可选的，没有特殊需求可以跳过
        cpf := profile.NewClientProfile()
        cpf.HttpProfile.Endpoint = "ams.tencentcloudapi.com"
        // 实例化要请求产品的client对象,clientProfile是可选的
        client, _ := ams.NewClient(credential, "", cpf)

        // 实例化一个请求对象,每个接口都会对应一个request对象
        request := ams.NewCreateAudioModerationTaskRequest()

        request.BizType = common.StringPtr("default")
        request.Type = common.StringPtr("AUDIO")
        request.Tasks = []*ams.TaskInput {
                &ams.TaskInput {
                        DataId: common.StringPtr("111"),
                        Name: common.StringPtr("111"),
                        Input: &ams.StorageInfo {
                                Type: common.StringPtr("URL"),
                                Url:
common.StringPtr("https://xxxxx.com.cn/EQjp8HObL2VLbxEx/fc/open/image/20
22092921/xxx.jpg"),
                        },
                }
        }

        // 返回的resp是一个CreateAudioModerationTaskResponse的实例，与请求对
象对应
        response, err := client.CreateAudioModerationTask(request)
        if _, ok := err.(*errors.TencentCloudSDKError); ok {
                fmt.Printf("An API error has returned: %s", err)
                return
        }
        if err != nil {
                panic(err)
        }
        // 输出json格式的字符串回包
        fmt.Printf("%s", response.ToJsonString())
}
```

热点问题

SDK 是否支持设置超时时间?

SDK 具有默认的超时时间，除非必要，请勿更改默认设置。如果需要修改，可以参考 SDK 仓库 的详细版示例。

详细版

```
import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.exception.TencentCloudSDKException;
// 导入对应产品模块的client
import com.tencentcloudapi.cvm.v20170312.CvmClient;
// 导入要请求接口对应的request response类
import com.tencentcloudapi.cvm.v20170312.models.DescribeInstancesRequest;
import com.tencentcloudapi.cvm.v20170312.models.DescribeInstancesResponse;
import com.tencentcloudapi.cvm.v20170312.models.Filter;
//导入可选配置类
import com.tencentcloudapi.common.profile.ClientProfile;
import com.tencentcloudapi.common.profile.HttpProfile;
import com.tencentcloudapi.common.profile.Language;

public class DescribeInstances {
    public static void main(String[] args) {
        try {
            // 实例化一个认证对象，入参需要传入腾讯云账户 SecretId，SecretKey。
            // 为了保护密钥安全，建议将密钥设置在环境变量中或者配置文件中，请参考本文凭证管理章节。
            // 硬编码密钥到代码中有可能随代码泄露而暴露，有安全隐患，并不推荐。
            // Credential cred = new Credential("SecretId", "SecretKey");
            Credential cred = new Credential(System.getenv("TENCENTCLOUD_SECRET_ID"), System.getenv(

            // 实例化一个http选项，可选的，没有特殊需求可以跳过
            HttpProfile httpProfile = new HttpProfile();
            // 从3.0.96版本开始，单独设置 HTTP 代理
            // httpProfile.setProxyHost("真实代理ip");
            // httpProfile.setProxyPort(真实代理端口);
            httpProfile.setReqMethod("GET"); // get请求(默认为post请求)
            httpProfile.setConnTimeout(30); // 请求连接超时时间，单位为秒(默认60秒)
            httpProfile.setWriteTimeout(30);  // 设置写入超时时间，单位为秒(默认0秒)
            httpProfile.setReadTimeout(30);  // 设置读取超时时间，单位为秒(默认0秒)
            httpProfile.setEndpoint("cvm.ap-shanghai.tencentcloudapi.com"); // 指定接入地域域名(默认就近
```

多线程是否可以共用一个 Client?

SDK 目前不支持线程安全，因此需要在每个线程中创建一个 Client 来进行请求。

同时连接数量是否有限制?

目前没有限制，但是接口会限制每秒请求数（QPS）的并发数，音频接口默认为20QPS。建议客户端也控制并发数，因为太多线程可能超出用户机器的处理能力，导致连接失败。

接入后验证

最近更新时间：2026-09-08 17:18:33

当您完成 API 或 SDK 接入后，可通过如下方式验证音频内容检测服务是否接入成功。

接入成功

若 Response 回参没有“Error”字段且正常返回业务参数，则说明 AMS 接口接入成功，示例如下所示：


```
{
    "Response": {
        "Results": [
            {
                "DataId": "0a782332-c9db-4cf5-a66e-20d60b4ea469",
                "TaskId": "c933aca1-90d2-4ab8-b045-f1b08069d76f",
                "Code": "OK",
                "Message": "Success"
            }
        ],
        "RequestId": "c933aca1-90d2-4ab8-b045-f1b08069d76f"
    }
}
```


异步回调结果示例

音频片段回调


```
{
    "TaskId": "w-audio-ZSS_LFTB3lSvDHmP",
    "DataId": "test",
    "BizType": "default",
    "Name": "",
    "Status": "RUNNING",
    "Type": "AUDIO",
    "Suggestion": "Block",
    "Labels": [],
    "InputInfo": null,
    "MediaInfo": null,
    "AudioText": "",
```

```
    "ImageSegments": [],
    "AudioSegments": [
        {
            "Result": {
                "HitFlag": 1,
                "Url": "https://cos.ap-
guangzhou.myqcloud.com/yunyingpingtai-xxxx/_cms_segments/cms/vod/w-
audio-ZSS_LFTB3lSvDHmP/audio_120_1696907088.mp3",
                "Suggestion": "Block",
                "Label": "Terror",
                "Text": "xxxxxxxxxxxxxxxxxxx",
                "TextResults": [
                    {
                        "Label": "Illegal",
                        "Score": 0,
                        "Keywords": [],
                        "Suggestion": "Pass",
                        "LibId": "",
                        "LibType": 0,
                        "LibName": "",
                        "SubLabel": ""
                    },
                    {
                        "Label": "Spam",
                        "Score": 0,
                        "Keywords": [],
                        "Suggestion": "Pass",
                        "LibId": "",
                        "LibType": 0,
                        "LibName": "",
                        "SubLabel": ""
                    },
                    {
                        "Label": "Terror",
                        "Score": 0,
                        "Keywords": [],
                        "Suggestion": "Block",
                        "LibId": "",
                        "LibType": 0,
                        "LibName": "",
```

```
                "SubLabel": "Crowd"
            },
            {
                "Label": "Porn",
                "Score": 2,
                "Keywords": [],
                "Suggestion": "Pass",
                "LibId": "",
                "LibType": 0,
                "LibName": "",
                "SubLabel": ""
            },
            {
                "Label": "Ad",
                "Score": 0,
                "Keywords": [],
                "Suggestion": "Pass",
                "LibId": "",
                "LibType": 0,
                "LibName": "",
                "SubLabel": ""
            },
            {
                "Label": "Abuse",
                "Score": 1,
                "Keywords": [],
                "Suggestion": "Pass",
                "LibId": "",
                "LibType": 0,
                "LibName": "",
                "SubLabel": ""
            }
        ],
        "MoanResults": [
            {
                "Label": "Moan",
                "Score": 1,
                "StartTime": 0,
                "EndTime": 15,
                "Suggestion": "Pass",
```

```
                        "SubLabel": ""
                    }
                ],
                "LanguageResults": [
                    {
                        "Label": "yue",
                        "Score": 99,
                        "StartTime": 0,
                        "EndTime": 15
                    }
                ],
                "Duration": "15000",
                "Score": 98,
                "Extra": "",
                "SpeakerResults": [],
                "LabelResults": [],
                "TravelResults": [],
                "SubLabel": "ReactionaryDivision",
                "RecognitionResults": []
            },
            "OffsetTime": "120",
            "CreatedAt": null
        }
    ],
    "TryInSeconds": 0,
    "CreatedAt": null,
    "UpdatedAt": null,
    "ErrorType": "",
    "ErrorDescription": "",
    "Asrs": [],
    "Label": "",
    "SegmentCosUrlList": null
}
```


审核结束回调


```
{
    "TaskId": "w-audio-ZSS_LFTB3lSvDHmP",
    "DataId": "test",
```

```
    "BizType": "default",
    "Name": "测试音频",
    "Status": "FINISH",
    "Type": "AUDIO",
    "Suggestion": "Block",
    "Labels": [
        {
            "Label": "Terror",
            "Suggestion": "Review",
            "Score": 99,
            "SubLabel": "Crowd"
        }
    ],
    "InputInfo": null,
    "MediaInfo": {
        "Codecs": "",
        "Duration": 144,
        "Width": 0,
        "Height": 0,
        "Thumbnail": ""
    },
    "AudioText": "xxxxxxxxxxxxxxxxxxx",
    "ImageSegments": [],
    "AudioSegments": [
        {
            "Result": {
                "HitFlag": 1,
                "Url": "https://cos.ap-
guangzhou.myqcloud.com/yunyingpingtai-xxxx/_cms_segments/cms/vod/w-
audio-ZSS_LFTB3lSvDHmP/audio_60_1696907073.mp3",
                "Suggestion": "Block",
                "Label": "Polity",
                "Text": "xxxxxxxxxxxxxxxxxxx",
                "TextResults": [
                    {
                        "Label": "Terror",
                        "Score": 0,
                        "Keywords": [],
                        "Suggestion": "Block",
                        "LibId": "",
```

```
                "LibType": 0,
                "LibName": "",
                "SubLabel": "Crowd"
            },
            {
                "Label": "Porn",
                "Score": 2,
                "Keywords": [],
                "Suggestion": "Pass",
                "LibId": "",
                "LibType": 0,
                "LibName": "",
                "SubLabel": ""
            },
            {
                "Label": "Ad",
                "Score": 0,
                "Keywords": [],
                "Suggestion": "Pass",
                "LibId": "",
                "LibType": 0,
                "LibName": "",
                "SubLabel": ""
            },
            {
                "Label": "Abuse",
                "Score": 0,
                "Keywords": [],
                "Suggestion": "Pass",
                "LibId": "",
                "LibType": 0,
                "LibName": "",
                "SubLabel": ""
            },
            {
                "Label": "Illegal",
                "Score": 0,
                "Keywords": [],
                "Suggestion": "Pass",
                "LibId": "",
```

```
                "LibType": 0,
                "LibName": "",
                "SubLabel": ""
            },
            {
                "Label": "Spam",
                "Score": 0,
                "Keywords": [],
                "Suggestion": "Pass",
                "LibId": "",
                "LibType": 0,
                "LibName": "",
                "SubLabel": ""
            }
        ],
        "MoanResults": [
            {
                "Label": "Moan",
                "Score": 3,
                "StartTime": 0,
                "EndTime": 15,
                "Suggestion": "Pass",
                "SubLabel": ""
            }
        ],
        "LanguageResults": [
            {
                "Label": "yue",
                "Score": 90,
                "StartTime": 0,
                "EndTime": 15
            },
            {
                "Label": "dialect",
                "Score": 9,
                "StartTime": 0,
                "EndTime": 15
            }
        ],
        "Duration": "15000",
```

```
                "Score": 99,
                "Extra": "",
                "SpeakerResults": [],
                "LabelResults": [],
                "TravelResults": [],
                "SubLabel": "NationalInstitution",
                "RecognitionResults": []
            },
            "OffsetTime": "60",
            "CreatedAt": "2023-10-10T11:04:34Z"
        }
    ],
    "TryInSeconds": 0,
    "CreatedAt": "1970-01-01T00:00:00Z",
    "UpdatedAt": null,
    "ErrorType": "",
    "ErrorDescription": "",
    "Asrs": [
        {
            "text": "xxxxxxxxxxxxxxxxxxx",
            "CreatedAt": "2023-10-10T11:04:34Z"
        },
        {
            "text": "xxxxxxxxxxxxxxxxxxx",
            "CreatedAt": "2023-10-10T11:04:38Z"
        }
    ],
    "Label": "Porn",
    "SegmentCosUrlList": null
}
```


接入失败

若 Response 回参显示“Error”字段，则说明接入失败，以下为 signature 校验失败的示例：


```
{
    "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"
    }
}
```

其中 Error 中的 Code 表示错误码，Message 表示错误的具体信息，详情请参见 错误码 。

回调配置说明

最近更新时间：2026-09-08 17:18:33

回调准备工作

确定回调 URL

要开始设置回调服务，请首先确定您要接收回调的 URL。这是您的系统将接收来自我们服务器的 HTTP POST 请求的地址。确保您的 URL 是公共可访问的，并支持 HTTPS 协议。

配置回调参数

在您的系统中，您需要配置一些参数以接收和处理我们发送的 HTTP POST 请求。以下是一些重要的参数：

- 请求方法：HTTP POST。
- 请求头：Content-Type: application / json。
- 请求体：JSON 格式的数据，包含有关回调结果的详细信息，详情信息请参考接收回调结果。

您需要根据您的系统需求调整这些参数。请确保您的系统能够解析和处理 JSON 格式的请求体。

测试回调服务

在实际使用回调服务之前，请确保您已经成功地设置了回调 URL 和参数，并且您的系统已经可以接收和处理来自我们服务器的 HTTP POST 请求。您可以使用一些在线工具来模拟 HTTP POST 请求，以确保您的回调服务正常工作。

处理回调事件

当我们的服务器发送 HTTP POST 请求到您的回调 URL 时，您的系统需要解析请求体中的 JSON 数据，并根据事件状态 Status 执行相应的操作。以下是一些常见的事件类型：

- FINISH：任务审核成功。
- RUNNING：任务还在执行中。
- ERROR：任务执行失败，可参考回调中的失败类型（ErrorType）和具体原因（ErrorDescription）字段来确认失败原因。

请根据您的系统需求来处理不同的事件类型。请确保响应的 HTTP 状态码为200 OK。

接收回调结果

请查阅 接入后验证 以获取回调示例，字段释义详见 查询任务详情 接口文档。

> ⓘ **注意：**
> 在点播场景中，默认情况下只有当审核结束（Finish）时才会触发回调。而在直播场景中，默认情况下只有在检测到违规和审核结束时才会触发回调。

RUNNING 输入参数

任务审核中，该状态会对单个图片或音频片段的检测结果进行回调。

FINISH 输入参数

任务审核结束，该状态下会返回前30个图片/音频违规片段结果，完整结果可通过 running 片段回调去拼接，或者调用查询任务详情接口来获取。

验签流程

为了验证接收到的回调数据是否来自合法的发送方并且在传输过程中未被篡改，用户在创建任务时可以选择向我们传递 seed 值，如果不需要验签则可以不用传递。在回调时，我们会加入签名参数以确保数据的安全性。我们将在返回的 HTTP 头部中添加 `X-Signature` 字段，其值将为 `seed` + `body` 的 sha256编码和 Hex 字符串。

ⓘ **例如：用户 CallbackUrl 是：** `http://example.com` **，Seed 是：** `dedb6dcc1cb7c63fde8fa5abfd57` **，我们返回的回调的数据是：**
{"TaskId": "task-video-X0zpcRUMzVidxj20","DataId":"test","Suggestion": "Block"}
则，审核完成后，我们会在调用 `http://example.com` **的时候，在 HTTP 头部 传入** `X-Signature` **的值为：** `74f0ae6d1f1e4eb1ffe4162da480a812f8a4dc19fe5a52bacbcd2c862d3edcfd`

```
#!/usr/bin/env python3
# -*- encoding: utf-8 -*-
# 验签示例

import hashlib

seed = "dedb6dcc1cb7c63fde8fa5abfd57"
body = '{"TaskId": "task-video-
X0zpcRUMzVidxj20","DataId":"test","Suggestion": "Block"}'
# 将 seed + body 转为 bytes 类型，并计算 SHA256 编码
sha256 = hashlib.sha256((seed + body).encode('utf-8')).hexdigest()
print(sha256)
```

回调 URL 续签

由于天御只具有临时访问用户 COS 桶的权限，临时权限最长只有 12 小时，因此，回调的 URL 有效期最长为 12 小时。建议在接收到回调时立即对 URL 进行签名更新和替换，以获取更长时效的链接。COS 目前支持的各语言 SDK 都提供了生成签名链接的功能，但获取长期时效链接需要使用 永久密钥 。以下提供了 java、python、go 对应 SDK 调用的示例。

说明：

回调 URL 格式： `https://<bucket>.cos.region.myqcloud.com/file_path`

- **回调 URL 示例：** `https://tianyu-live-video-content-moderation-xxxx.cos.ap-guangzhou.myqcloud.com/segment-/audio/w-live_video-YrvumJyVBc3KM1DF/1656483481.mp3`
- **续签后 URL 示例：** `https://tianyu-live-video-content-moderation-xxxx.cos.ap-guangzhou.myqcloud.com/segment-/audio/w-live_video-YrvumJyVBc3KM1DF/1656483481.mp3?q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=1754477817;1755082617&q-key-time=1754477817;1755082617&q-header-list=host&q-url-param-list=&q-signature=1ea700f9e6431787d22c5ecc6a6b4a1347909d91`

Python

```python
#!/usr/bin/env python3
# -*- coding:utf-8 -*-

from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client

# 密钥
secret_id = "" # 点击永久密钥获取
secret_key = "" # 点击永久密钥获取
region = "" # 默认 ap-guangzhou


def get_cos_presigned_url(url):
    """
    获取.....
    :param url:
    :return: presigned_url
    """
    # 获取客户端对象
    config = CosConfig(Region=region, SecretId=secret_id,
SecretKey=secret_key)
    client = CosS3Client(config)
    # 根据key获取下载链接
    # 获取签名的cos桶
```

```
    bucket_name = "tianyu-live-video-content-moderation-xxxx"
    # print(bucket_name)
    # 获取签名的 cos 路径，请确保路径不被编码
    key = "segment-/audio/w-live_video-YrvumJyVBc3KM1DF/1656483481.mp3"
    # print(key)
    # 使用永久密钥时，expired_sec 签名过期时间可以随意设置
    expired_sec = 60 * 60 * 24 * 7
    download_response = client.get_presigned_download_url(
        Bucket=bucket_name, Key=key, Expired=expired_sec)
    return download_response


if __name__ == '__main__':
    # presigned_url既为获得新的签名的URL
    presigned_url = get_cos_presigned_url(
        "url")
    print(presigned_url)
```

Java

```
import com.qcloud.cos.COSClient;
import com.qcloud.cos.ClientConfig;
import com.qcloud.cos.auth.BasicCOSCredentials;
import com.qcloud.cos.auth.COSCredentials;
import com.qcloud.cos.http.HttpMethodName;
import com.qcloud.cos.http.HttpProtocol;
import com.qcloud.cos.region.Region;
import java.net.URL;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

public class TencentCosClient{
    public static void main(String [] args){
        // 调用 cos 接口之前必须保证本进程存在一个 COSClient 实例，如果没有则创建
        // 详细代码参见本页：简单操作 -> 创建 COSClient
        COSClient cosClient = createCOSClient();

        // 获取签名的cos桶
```

```
        String bucketName = "tianyu-live-video-content-moderation-xxxx";
        // 获取签名的 COS 路径，请确保路径不被编码
        String key = "/segment-/audio/w-live_video-
YrvumJyVBc3KM1DF/1656483481.mp3";

        // 设置签名过期时间(可选)， 若未进行设置则默认使用 ClientConfig 中的签名过
期时间(1小时)
        // 这里设置签名在半个小时后过期
        Date expirationDate = new Date(System.currentTimeMillis() + 30 *
60 * 1000);

        // 填写本次请求的参数，需与实际请求相同，能够防止用户篡改此签名的 HTTP 请求
的参数
        Map<String, String> params = new HashMap<String, String>();
        params.put("param1", "value1");

        // 填写本次请求的头部，需与实际请求相同，能够防止用户篡改此签名的 HTTP 请求
的头部
        Map<String, String> headers = new HashMap<String, String>();
        headers.put("header1", "value1");

        // 请求的 HTTP 方法，上传请求用 PUT，下载请求用 GET，删除请求用 DELETE
        HttpMethodName method = HttpMethodName.GET;

        URL url = cosClient.generatePresignedUrl(bucketName, key,
expirationDate, method, headers, params);
        System.out.println(url.toString());

        // 确认本进程不再使用 cosClient 实例之后，关闭之
        cosClient.shutdown();
    }
    // 创建 COSClient 实例，这个实例用来后续调用请求
    public static COSClient createCOSClient() {
        // 设置用户身份信息。
        // SECRETID 和 SECRETKEY 请登录访问管理控制台
https://console.cloud.tencent.com/cam/capi 进行查看和管理
        String secretId = "";
        String secretKey = "";
        COSCredentials cred = new BasicCOSCredentials(secretId,
secretKey);
```

```
        // ClientConfig 中包含了后续请求 COS 的客户端设置：
        ClientConfig clientConfig = new ClientConfig();

        // 设置 bucket 的地域
        // COS_REGION 请参照
https://cloudexpirationDate.tencent.com/document/product/436/6224
        clientConfig.setRegion(new Region("ap-guangzhou"));

        // 设置请求协议, http 或者 https
        // 5.6.53 及更低的版本，建议设置使用 https 协议
        // 5.6.54 及更高版本，默认使用了 https
        clientConfig.setHttpProtocol(HttpProtocol.https);

        // 以下的设置，是可选的：
        // 设置 socket 读取超时，默认 30s
        clientConfig.setSocketTimeout(30*1000);
        // 设置建立连接超时，默认 30s
        clientConfig.setConnectionTimeout(30*1000);

        // 如果需要的话，设置 http 代理，ip 以及 port
        clientConfig.setHttpProxyIp("httpProxyIp");
        clientConfig.setHttpProxyPort(80);

        // 生成 cos 客户端。
        return new COSClient(cred, clientConfig);
    }
}
```

Go

```
package main

import (
    "context"
    "fmt"
    "net/http"
    "net/url"
    "time"
```

```
    "github.com/tencentyun/cos-go-sdk-v5"
)

// 通过tag的方式，用户可以将请求参数或者请求头部放进签名中。
type URLToken struct {
    SessionToken string `url:"x-cos-security-token,omitempty" header:"-
"`
}

func GetCosUrl() {
    // 替换成您的密钥
    tak := ""
    tsk := ""

    tmpURL := "https://tianyu-live-video-content-moderation-
xxxx/segment-/audio/w-live_video-YrvumJyVBc3KM1DF/1656483481.mp3?X-Amz-
Algorithmxxxx"

    parse, err := url.Parse(tmpURL)
    if err != nil {
        fmt.Println(err)
        return
    }
    // 获取签名的cos桶 tianyu-live-video-content-moderation-xxxx
    bucket_name := parse.Scheme + "://" + parse.Host
    // 获取签名的 COS 路径，请确保路径不被编码 /segment-/audio/w-live_video-
YrvumJyVBc3KM1DF/1656483481.mp3
    key := parse.Path

    u, _ := url.Parse(bucket_name)
    b := &cos.BaseURL{BucketURL: u}
    c := cos.NewClient(b, &http.Client{})

    name := key
    ctx := context.Background()

    // 通过 tag 设置 x-cos-security-token
    // Get presigned
```

```
    presignedURL, err := c.Object.GetPresignedURL(ctx, http.MethodGet,
name, tak, tsk, 100*time.Hour, nil)
    if err != nil {
        fmt.Printf("Error: %v\n", err)
        return
    }
    // Get object by presinged url
    resp, err := http.Get(presignedURL.String())
    if err != nil {
        fmt.Printf("Error: %v\n", err)
    }
    defer resp.Body.Close()
    fmt.Println(presignedURL.String())
    fmt.Printf("resp:%v\n", resp)
}
```

回调异常热点问题

未收到回调结果一般是什么原因?

1. 任务还在 Pending 中，可以查询任务详情接口看 Status 状态。任务审核逻辑是新增任务优先审核，旧任务往后排队。
2. 当配置为最终结果回调时，只有在审核任务 Finish 结束后才会触发回调。您可以通过详情接口查看 Status 任务状态，也可以通过控制台修改回调逻辑，以支持片段和最终结果的回调。
3. 用户侧未按照正确的响应码响应200，导致审核回调失败，可以使用类似 Postman 或 curl 等工具发送请求，并查看响应代码是否为200。
4. 回调等待响应头时超时，回调超时时间超过5s 会报错"context deadline exceeded (Client.Timeout exceeded while awaiting headers)" ，该情况需要提供 taskId 给到产品侧确认。
    - 检查回调服务是否正常运行，并确保其能够及时响应请求。
    - 检查网络连接是否稳定，并尝试使用更快的网络连接。
    - 如果请求被防火墙或其他网络安全设备拦截，可以尝试修改其配置，以允许请求通过。

建议客户侧可以加上调试和监控功能系统。可以使用日志记录来记录请求和响应数据，以便进行故障排除。您还可以使用监控工具来监视您的回调服务的性能和可用性。

回调 URL 为什么无法打开?

如果用户的接收程序对回调的内容进行了编码，会导致链接无法正确打开。

视频内容安全接入指引
API 接入指引

最近更新时间：2026-09-08 17:18:33

腾讯云 API 会对每个请求进行身份验证，用户使用原生 API 接入需要使用安全凭证，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，以下内容主要介绍 API 接口文档参数以及 V3签名方法。

API 文档

参考 视频内容安全 接口文档，了解请求 API 所需要的公共参数、域名、接口输入输出参数值。

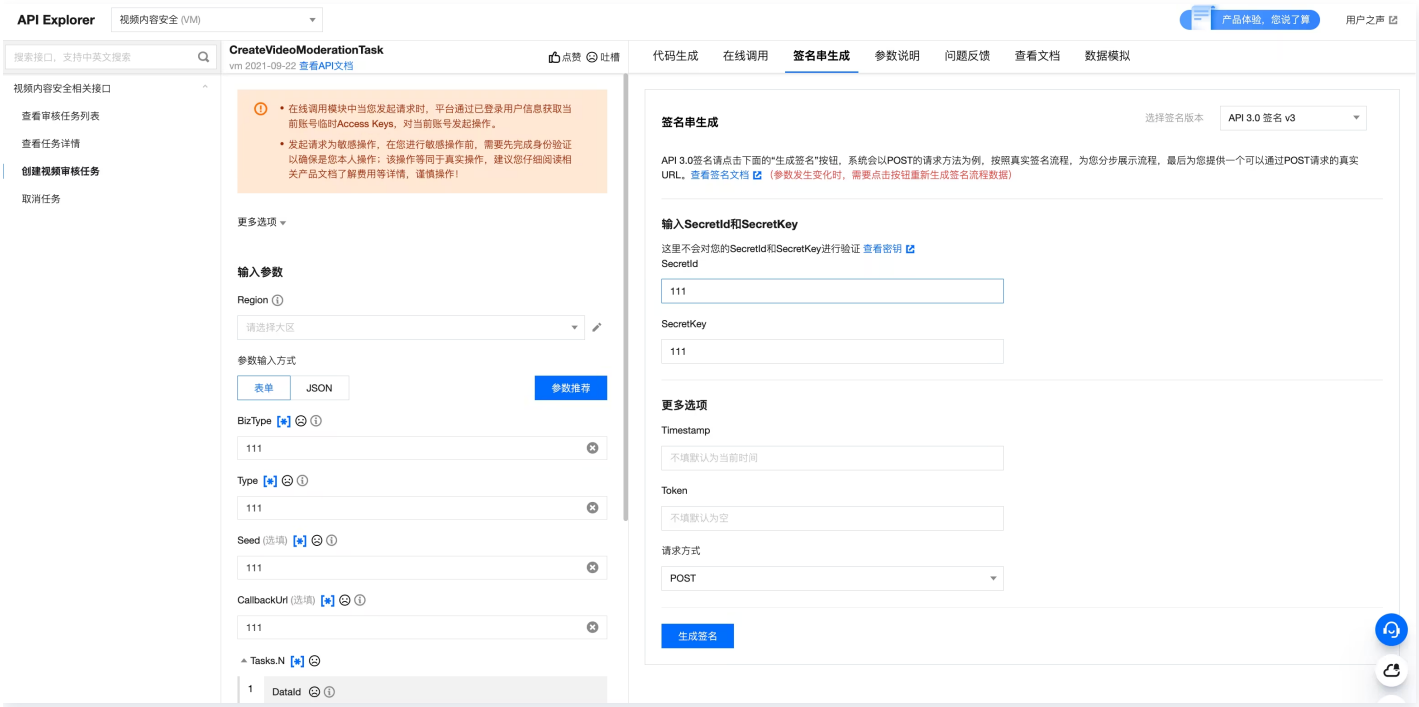
公共参数

公共参数是用于标识用户和接口签名的参数，API 接口文档上不会展示公共参数说明，但每次请求 API 均需要携带这些参数，才能正常发起请求。

签名方法

腾讯云支持 V1、V3两种签名方式，推荐使用签名方法 V3计算签名，签名方法 V3（或被称为 TC3-HMAC-SHA256）相比签名方法 V1更安全，支持更大的请求包，且 POST JSON 格式，性能有一定提升。

首次接触，建议使用 API Explorer 中的“签名串生成”功能，选择签名版本为“API 3.0 签名 V3”，可以生成签名过程进行验证。



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，该字段取值为 CreateVideoModerationTask。
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。该字段取值为 2020-12-29。
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 vm；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，计算过程请参见 签名版本 v3 签名过程 。
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。

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` 协议格式。

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，POST 请求支持 10MB 以内的请求包，签名过程详见 文档 。

示例代码

以下为 Java、Go、Python 示例 demo，填写密钥信息、接口入参即可调用，其他语言请参考 签名方法 V3-签名演示 ，签名演示是云服务器服务作为示例，**实际调用时需要根据 API 接口文档填写参数**。

Java

```
import java.nio.charset.Charset;
import java.io.OutputStream;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
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 CreateVideoModeration {
    private final static Charset UTF8 = StandardCharsets.UTF_8;
    private final static String SECRET_ID = "";
    private final static String 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 = "vm";
        String host = "vm.tencentcloudapi.com";
        String region = "ap-guangzhou";
        String action = "CreateVideoModeration";
        String version = "2021-09-22";
        String algorithm = "TC3-HMAC-SHA256";
        String timestamp = String.valueOf(System.currentTimeMillis() /
1000);
```

```
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
        String date = sdf.format(new Date(Long.valueOf(timestamp +
"000")));

        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 = "{\"Type\": \"xxxx\",\"BizType\": \"xxxx\"}";//
body数据为json
        String hashedRequestPayload = sha256Hex(payload);
        String canonicalRequest = httpRequestMethod + "\n" +
canonicalUri + "\n" + canonicalQueryString + "\n"
                + canonicalHeaders + "\n" + signedHeaders + "\n" +
hashedRequestPayload;

        String credentialScope = date + "/" + service + "/" +
"tc3_request";
        String hashedCanonicalRequest = sha256Hex(canonicalRequest);
        String stringToSign = algorithm + "\n" + timestamp + "\n" +
credentialScope + "\n" + hashedCanonicalRequest;

        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();

        String authorization = algorithm + " " + "Credential=" +
SECRET_ID + "/" + credentialScope + ", "
                + "SignedHeaders=" + signedHeaders + ", " + "Signature="
+ signature;
```

```
        try {
            URL url = new URL("https://" + host);
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            connection.setRequestMethod("POST");
            connection.setRequestProperty("Authorization",
authorization);
            connection.setRequestProperty("Content-Type", CT_JSON);
            connection.setRequestProperty("Host", host);
            connection.setRequestProperty("X-TC-Action", action);
            connection.setRequestProperty("X-TC-Timestamp", timestamp);
            connection.setRequestProperty("X-TC-Version", version);
            connection.setRequestProperty("X-TC-Region", region);

            // Enable input/output streams and write the payload
            connection.setDoOutput(true);
            OutputStream os = connection.getOutputStream();
            os.write(payload.getBytes(UTF8));
            os.flush();
            os.close();

            // Get the response
            int responseCode = connection.getResponseCode();
            if (responseCode == 200) {
                BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
                String inputLine;
                StringBuilder response = new StringBuilder();

                while ((inputLine = in.readLine()) != null) {
                    response.append(inputLine);
                }
                in.close();

                // Process the response as needed
                System.out.println("Response: " + response.toString());
            } else {
                // Handle the error response
```

```
                System.out.println("Error Response Code: " +
responseCode);
            }

            connection.disconnect();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Go

```
package vm

import (
    "crypto/hmac"
    "crypto/sha256"
    "encoding/hex"
    "encoding/json"
    "fmt"
    "io/ioutil"
    "net/http"
    "strconv"
    "strings"
    "time"
)

type createmod struct {
    BizType string `json:"BizType"`
    Type    string `json:"Type"`
    Tasks   []task
}
type task struct {
    DataId string    `json:"DataId"`
    Input  taskInput `json:"Input"`
}

type taskInput struct {
```

```
    Type string `json:"Type"`
    Url  string `json:"Url"`
}

func sha256heCreate(s string) string {
    b := sha256.Sum256([]byte(s))
    return hex.EncodeToString(b[:])
}

func hmacsha256Create(s, key string) string {
    hashed := hmac.New(sha256.New, []byte(key))
    hashed.Write([]byte(s))
    return string(hashed.Sum(nil))
}

func DetectCreateSingle() {
    secretId := ""
    secretKey := ""
    host := "vm.tencentcloudapi.com"
    algorithm := "TC3-HMAC-SHA256"
    service := "vm"
    version := "2021-09-22"
    action := "CreateVideoModerationTask"
    region := "ap-guangzhou"
    var timestamp int64 = time.Now().Unix()
    // loc, _ := time.LoadLocation("Asia/Shanghai")
    // var timestamp int64 = time.Now().In(loc).Unix()

    // step 1: build canonical request string
    httpRequestMethod := "POST"
    canonicalURI := "/"
    canonicalQueryString := ""
    canonicalHeaders := "content-type:application/json; charset=utf-8\n"
+ "host:" + host + "\n"
    signedHeaders := "content-type;host"
    var a = createmod{
        BizType: "default",
        Type:    "VIDEO",
        Tasks: []task{
            {
```

```
                DataId: "123",
                Input: taskInput{
                    Type: "URL",
                    Url:  "", //接口入参
                },
            },
        },
    }
    b, err := json.Marshal(a)
    if err != nil {
        fmt.Print(err.Error())
        return
    }
    payload := string(b)
    hashedRequestPayload := sha256heCreate(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 := sha256heCreate(canonicalRequest)
    string2sign := fmt.Sprintf("%s\n%d\n%s\n%s",
        algorithm,
        timestamp,
        credentialScope,
        hashedCanonicalRequest)
    // fmt.Println(string2sign)

    // step 3: sign string
    secretDate := hmacsha256Create(date, "TC3"+secretKey)
    secretService := hmacsha256Create(service, secretDate)
    secretSigning := hmacsha256Create("tc3_request", secretService)
```

```
    signature := hex.EncodeToString([]byte(hmacsha256Create(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)

    req, _ := http.NewRequest("POST", "https://"+host,
strings.NewReader(payload))
    req.Header.Set("Authorization", authorization)
    req.Header.Set("Content-Type", "application/json; charset=utf-8")
    req.Header.Set("Host", host)
    req.Header.Set("X-TC-Action", action)
    req.Header.Set("X-TC-Timestamp", strconv.FormatInt(timestamp, 10))
    req.Header.Set("X-TC-Version", version)
    req.Header.Set("X-TC-Region", region)

    resp, err := (&http.Client{}).Do(req)
    if err != nil {
        fmt.Print(err.Error())
        return
    }
    defer resp.Body.Close()

    respByte, _ := ioutil.ReadAll(resp.Body)
    fmt.Println(string(respByte))
}
```

Python

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import hashlib
```

```
import hmac
import json
import time
import datetime
import requests

from concurrent.futures import ThreadPoolExecutor

secret_id = ""
secret_key = ""

service = "vm"
host = "vm.tencentcloudapi.com"
endpoint = "https://" + host
region = "ap-guangzhou"
version = "2021-09-22"
algorithm = "TC3-HMAC-SHA256"


def do_action(action, params):

    timestamp = int(time.time())
    # datetime.datetime.utcfromtimestamp
    day = datetime.datetime.utcfromtimestamp(timestamp).strftime("%Y-%m-
%d")

    # ************* 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)
    # print(payload)
    canonical_headers = "content-type:%s\nhost:%s\n" % (ct, host)
    signed_headers = "content-type;host"
    hashed_request_payload = hashlib.sha256(
        payload.encode("utf-8")).hexdigest()
    # print(hashed_request_payload)
    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 = day + "/" + 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)

    secret_date = sign(("TC3" + secret_key).encode("utf-8"), day)
    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)

    headers = {
        "Authorization": authorization,
        "Content-Type": "application/json; charset=utf-8",
        "Host": host,
        "X-TC-Action": action,
        "X-TC-Timestamp": str(timestamp),
        "X-TC-Version": version,
        "X-TC-Region": region
    }
    # time.sleep(15)
    # 发送请求
```

```
    resp = requests.post(url=endpoint, data=payload, headers=headers)

    return resp


# 计算签名摘要函数
def sign(key, msg):
    return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()


def audio_detect(params):
    action = "CreateVideoModerationTask"
    resp = do_action(action, params)

    return resp.text


if __name__ == "__main__":
    params = []
    video_url = ""
    p = {
        "Type":
        "VIDEO",
        "Tasks": [{
            "DataId": "default",
            "Name": "测试视频",
            "Input": {
                "Type": "URL",
                "Url": video_url
            }
        }],
        "BizType":
        "default"
    }
    params.append(p)
    executor = ThreadPoolExecutor(max_workers=5)
    resps = executor.map(audio_detect, params)
    for resp in resps:
        print(resp)
```

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

SDK 接入指引

最近更新时间：2026-09-08 17:18:33

推荐您前往使用腾讯云 API 配套的 7 种常见的编程语言 SDK，已经封装了签名和请求过程，开发时只关注产品提供的具体接口即可。

API 文档

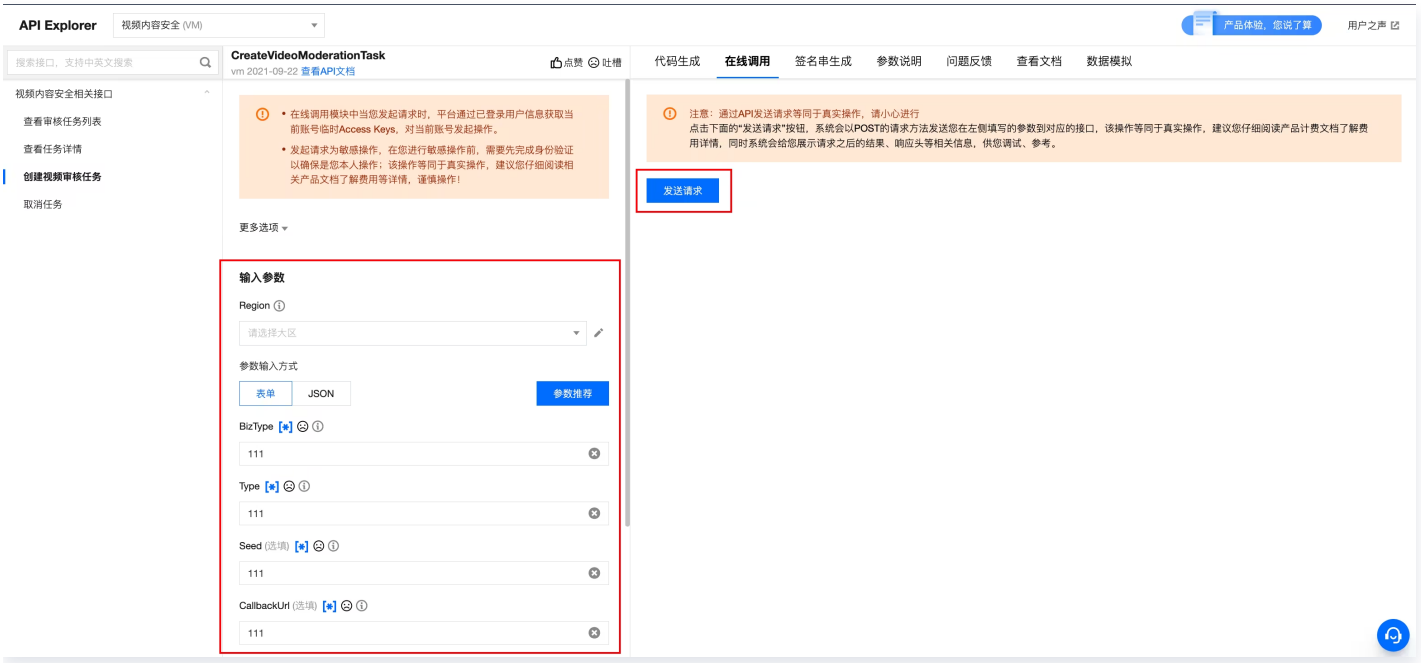
参考 视频内容安全 接口文档，了解需要使用的参数。

SDK 使用说明

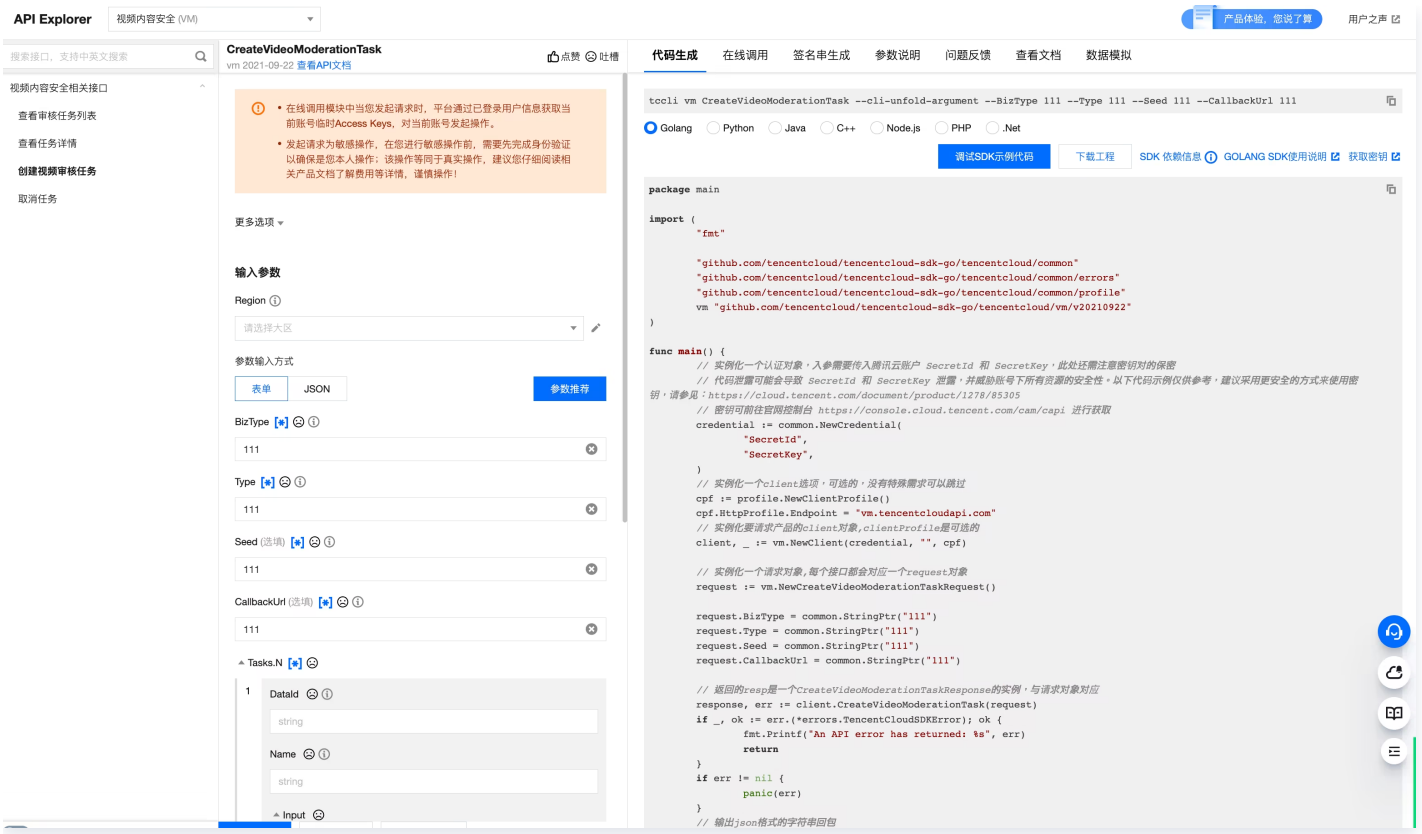
云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，均已开源，能更方便的调用 API，支持 Python 、Java 、PHP 、Go 、NodeJS 、.NET 、C++ 。
各语言 SDK 使用说明可参考 文档，介绍依赖版本以及安装过程等。

建议安装最新版本的 SDK，即可调用接口，腾讯云提供了 API Explorer 可在线调用，并且可以快速生成 SDK 调用示例代码，以下 Java、Python、Golang 示例均从 API Explorer 生成，其他语言可自行调试获取。

在线调用示例



代码生成示例



Java

```java
import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.profile.ClientProfile;
import com.tencentcloudapi.common.profile.HttpProfile;
import com.tencentcloudapi.common.exception.TencentCloudSDKException;
import com.tencentcloudapi.vm.v20210922.VmClient;
import com.tencentcloudapi.vm.v20210922.models.*;

public class CreateVideoModerationTask
{
    public static void main(String [] args) {
        try{
            // 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处还需注意密钥对的保密
            // 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全性。以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见：https://cloud.tencent.com/document/product/1278/85305
            // 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi 进行获取
            Credential cred = new Credential("SecretId", "SecretKey");
```

```
            // 实例化一个http选项，可选的，没有特殊需求可以跳过
            HttpProfile httpProfile = new HttpProfile();
            httpProfile.setEndpoint("vm.tencentcloudapi.com");
            // 实例化一个client选项，可选的，没有特殊需求可以跳过
            ClientProfile clientProfile = new ClientProfile();
            clientProfile.setHttpProfile(httpProfile);
            // 实例化要请求产品的client对象,clientProfile是可选的
            VmClient client = new VmClient(cred, "ap-guangzhou",
clientProfile);
            // 实例化一个请求对象,每个接口都会对应一个request对象
            CreateVideoModerationTaskRequest req = new
CreateVideoModerationTaskRequest();
            req.setBizType("default");
            req.setType("VIDEO");

            TaskInput[] taskInputs1 = new TaskInput[1];
            TaskInput taskInput1 = new TaskInput();
            taskInput1.setDataId("test");
            taskInput1.setName("test");
            StorageInfo storageInfo1 = new StorageInfo();
            storageInfo1.setType("URL");
            storageInfo1.setUrl("http://xxx.mp4");
            taskInput1.setInput(storageInfo1);

            taskInputs1[0] = taskInput1;

            req.setTasks(taskInputs1);

            // 返回的resp是一个CreateVideoModerationTaskResponse的实例，与请
求对象对应
            CreateVideoModerationTaskResponse resp =
client.CreateVideoModerationTask(req);
            // 输出json格式的字符串回包

System.out.println(CreateVideoModerationTaskResponse.toJsonString(resp))
;
        } catch (TencentCloudSDKException e) {
            System.out.println(e.toString());
        }
    }
```

```
}
```

Python

```
import json
from tencentcloud.common import credential
from tencentcloud.common.profile.client_profile import ClientProfile
from tencentcloud.common.profile.http_profile import HttpProfile
from tencentcloud.common.exception.tencent_cloud_sdk_exception import
TencentCloudSDKException
from tencentcloud.vm.v20210922 import vm_client, models
try:
    # 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处还需注
意密钥对的保密
    # 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全
性。以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见:
https://cloud.tencent.com/document/product/1278/85305
    # 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi 进行获
取
    cred = credential.Credential("SecretId", "SecretKey")
    # 实例化一个http选项，可选的，没有特殊需求可以跳过
    httpProfile = HttpProfile()
    httpProfile.endpoint = "vm.tencentcloudapi.com"

    # 实例化一个client选项，可选的，没有特殊需求可以跳过
    clientProfile = ClientProfile()
    clientProfile.httpProfile = httpProfile
    # 实例化要请求产品的client对象,clientProfile是可选的
    client = vm_client.VmClient(cred, "ap-guangzhou", clientProfile)

    # 实例化一个请求对象,每个接口都会对应一个request对象
    req = models.CreateVideoModerationTaskRequest()
    params = {
        "BizType": "default",
        "Type": "VIDEO",
        "Tasks": [
            {
                "DataId": "test",
                "Name": "test",
```

```
                "Input": {
                    "Type": "URL",
                    "Url": "http://xxx.mp4"
                }
            }
        ]
    }
    req.from_json_string(json.dumps(params))

    # 返回的resp是一个CreateVideoModerationTaskResponse的实例，与请求对象对应
    resp = client.CreateVideoModerationTask(req)
    # 输出json格式的字符串回包
    print(resp.to_json_string())

except TencentCloudSDKException as err:
    print(err)
```

Go

```
package main

import (
        "fmt"

        "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common"
        "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/errors"
        "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/profile"
        vm "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/vm/v20210922"
)

func main() {
        // 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处
还需注意密钥对的保密
        // 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的
安全性。以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见：
```

```
https://cloud.tencent.com/document/product/1278/85305
        // 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi
进行获取
        credential := common.NewCredential(
                "SecretId",
                "SecretKey",
        )
        // 实例化一个client选项，可选的，没有特殊需求可以跳过
        cpf := profile.NewClientProfile()
        cpf.HttpProfile.Endpoint = "vm.tencentcloudapi.com"
        // 实例化要请求产品的client对象,clientProfile是可选的
        client, _ := vm.NewClient(credential, "ap-guangzhou", cpf)

        // 实例化一个请求对象,每个接口都会对应一个request对象
        request := vm.NewCreateVideoModerationTaskRequest()

        request.BizType = common.StringPtr("default")
        request.Type = common.StringPtr("VIDEO")
        request.Tasks = []*vm.TaskInput {
                &vm.TaskInput {
                        DataId: common.StringPtr("test"),
                        Name: common.StringPtr("test"),
                        Input: &vm.StorageInfo {
                                Type: common.StringPtr("URL"),
                                Url: common.StringPtr("http://xxx.mp4"),
                        },
                },
        }

        // 返回的resp是一个CreateVideoModerationTaskResponse的实例，与请求对
象对应
        response, err := client.CreateVideoModerationTask(request)
        if _, ok := err.(*errors.TencentCloudSDKError); ok {
                fmt.Printf("An API error has returned: %s", err)
                return
        }
        if err != nil {
                panic(err)
        }
        // 输出json格式的字符串回包
```

```
        fmt.Printf("%s", response.ToJsonString())
}
```

热点问题

SDK 是否支持设置超时时间?

SDK 具有默认的超时时间，除非必要，请勿更改默认设置。如果需要修改，可以参考 SDK 仓库 中的详细版示例。

详细版

```
import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.exception.TencentCloudSDKException;
// 导入对应产品模块的client
import com.tencentcloudapi.cvm.v20170312.CvmClient;
// 导入要请求接口对应的request response类
import com.tencentcloudapi.cvm.v20170312.models.DescribeInstancesRequest;
import com.tencentcloudapi.cvm.v20170312.models.DescribeInstancesResponse;
import com.tencentcloudapi.cvm.v20170312.models.Filter;
//导入可选配置类
import com.tencentcloudapi.common.profile.ClientProfile;
import com.tencentcloudapi.common.profile.HttpProfile;
import com.tencentcloudapi.common.profile.Language;

public class DescribeInstances {
    public static void main(String[] args) {
        try {
            // 实例化一个认证对象，入参需要传入腾讯云账户 SecretId，SecretKey。
            // 为了保护密钥安全，建议将密钥设置在环境变量中或者配置文件中，请参考本文凭证管理章节。
            // 硬编码密钥到代码中有可能随代码泄露而暴露，有安全隐患，并不推荐。
            // Credential cred = new Credential("SecretId", "SecretKey");
            Credential cred = new Credential(System.getenv("TENCENTCLOUD_SECRET_ID"), System.getenv(

            // 实例化一个http选项，可选的，没有特殊需求可以跳过
            HttpProfile httpProfile = new HttpProfile();
            // 从3.0.96版本开始，单独设置 HTTP 代理
            // httpProfile.setProxyHost("真实代理ip");
            // httpProfile.setProxyPort(真实代理端口);
            httpProfile.setReqMethod("GET"); // get请求(默认为post请求)
            httpProfile.setConnTimeout(30); // 请求连接超时时间，单位为秒(默认60秒)
            httpProfile.setWriteTimeout(30);  // 设置写入超时时间，单位为秒(默认0秒)
            httpProfile.setReadTimeout(30);  // 设置读取超时时间，单位为秒(默认0秒)
            httpProfile.setEndpoint("cvm.ap-shanghai.tencentcloudapi.com"); // 指定接入地域域名(默认就近
```

多线程是否可以共用一个 Client?

SDK 目前不支持线程安全，因此需要在每个线程中创建一个 Client 来进行请求。

同时连接数量是否有限制?

目前没有限制，但是接口会限制每秒请求数（QPS）的并发数，视频接口默认为1000QPS。建议客户端也控制并发数，因为太多线程可能超出用户机器的处理能力，导致连接失败。

接入后验证

最近更新时间：2026-09-08 17:18:33

当您完成 API 或 SDK 接入后，可以通过如下方式验证视频内容检测服务是否接入成功。

接入成功

若 Response 回参没有“Error”字段且正常返回业务参数，则说明 VM 接口接入成功，示例如下所示：


```
{
    "Response": {
        "Results": [
            {
                "DataId": "0a782332-c9db-4cf5-a66e-20d60b4ea469",
                "TaskId": "c933aca1-90d2-4ab8-b045-f1b08069d76f",
                "Code": "OK",
                "Message": "Success"
            }
        ],
        "RequestId": "c933aca1-90d2-4ab8-b045-f1b08069d76f"
    }
}
```


异步回调结果示例

图片片段回调


```
{
    "TaskId": "w-video-ZSS_LFTB3lSvDHmP",
    "DataId": "test",
    "BizType": "default",
    "Name": "",
    "Status": "RUNNING",
    "Type": "VIDEO",
    "Suggestion": "Block",
    "Labels": [],
    "InputInfo": null,
    "MediaInfo": null,
    "AudioText": "",
```

```
    "ImageSegments": [
        {
            "Result": {
                "HitFlag": 1,
                "Label": "Porn",
                "Suggestion": "Block",
                "Score": 91,
                "Results": [
                    {
                        "Scene": "Porn",
                        "HitFlag": 1,
                        "Suggestion": "Block",
                        "Label": "Porn",
                        "SubLabel": "PornSum",
                        "Score": 91,
                        "Names": [],
                        "Text": "",
                        "Details": []
                    },
                    {
                        "Scene": "OCR",
                        "HitFlag": 0,
                        "Suggestion": "Pass",
                        "Label": "Normal",
                        "SubLabel": "",
                        "Score": 0,
                        "Names": [],
                        "Text": "xxxxxxxxxxxxxxxxxxx",
                        "Details": [
                            {
                                "Name": "",
                                "Text": "xxxxxxxxxxxxxxxxxxx",
                                "Location": {
                                    "X": 0,
                                    "Y": 0,
                                    "Width": 0,
                                    "Height": 0,
                                    "Rotate": 0
                                },
                                "Label": "Normal",
```

```
                                "LibId": "",
                                "LibName": "",
                                "Keywords": [],
                                "Suggestion": "Pass",
                                "Score": 0,
                                "SubLabelCode": "",
                                "SubLabel": ""
                            }
                        ]
                    }
                ],
                "Url": "https://cos.ap-
guangzhou.myqcloud.com/yunyingpingtai-xxxxxxx/_cms_segments/cms/vod/w-
video-ZSS_LFTB3lSvDHmP/screenshot_135_1696907088.jpg",
                "Extra": "",
                "SubLabel": "PornSum",
                "RecognitionResults": []
            },
            "OffsetTime": "135",
            "CreatedAt": null,
            "OffsetusTime": "135"
        }
    ],
    "AudioSegments": [],
    "TryInSeconds": 0,
    "CreatedAt": null,
    "UpdatedAt": null,
    "ErrorType": "",
    "ErrorDescription": "",
    "Asrs": [],
    "Label": "",
    "SegmentCosUrlList": null
}
```


音频片段回调


```
{
    "TaskId": "w-audio-ZSS_LFTB3lSvDHmP",
    "DataId": "test",
```

```
    "BizType": "default",
    "Name": "",
    "Status": "RUNNING",
    "Type": "AUDIO",
    "Suggestion": "Block",
    "Labels": [],
    "InputInfo": null,
    "MediaInfo": null,
    "AudioText": "",
    "ImageSegments": [],
    "AudioSegments": [
        {
            "Result": {
                "HitFlag": 1,
                "Url": "https://cos.ap-
guangzhou.myqcloud.com/yunyingpingtai-xxxx/_cms_segments/cms/vod/w-
audio-ZSS_LFTB3lSvDHmP/audio_120_1696907088.mp3",
                "Suggestion": "Block",
                "Label": "Terror",
                "Text": "xxxxxxxxxxxxxxxxxxx",
                "TextResults": [
                    {
                        "Label": "Illegal",
                        "Score": 0,
                        "Keywords": [],
                        "Suggestion": "Pass",
                        "LibId": "",
                        "LibType": 0,
                        "LibName": "",
                        "SubLabel": ""
                    },
                    {
                        "Label": "Spam",
                        "Score": 0,
                        "Keywords": [],
                        "Suggestion": "Pass",
                        "LibId": "",
                        "LibType": 0,
                        "LibName": "",
                        "SubLabel": ""
```

```
        },
        {
            "Label": "Terror",
            "Score": 0,
            "Keywords": [],
            "Suggestion": "Block",
            "LibId": "",
            "LibType": 0,
            "LibName": "",
            "SubLabel": "Crowd"
        },
        {
            "Label": "Porn",
            "Score": 2,
            "Keywords": [],
            "Suggestion": "Pass",
            "LibId": "",
            "LibType": 0,
            "LibName": "",
            "SubLabel": ""
        },
        {
            "Label": "Ad",
            "Score": 0,
            "Keywords": [],
            "Suggestion": "Pass",
            "LibId": "",
            "LibType": 0,
            "LibName": "",
            "SubLabel": ""
        },
        {
            "Label": "Abuse",
            "Score": 1,
            "Keywords": [],
            "Suggestion": "Pass",
            "LibId": "",
            "LibType": 0,
            "LibName": "",
            "SubLabel": ""
```

```
                    }
                ],
                "MoanResults": [
                    {
                        "Label": "Moan",
                        "Score": 1,
                        "StartTime": 0,
                        "EndTime": 15,
                        "Suggestion": "Pass",
                        "SubLabel": ""
                    }
                ],
                "LanguageResults": [
                    {
                        "Label": "yue",
                        "Score": 99,
                        "StartTime": 0,
                        "EndTime": 15
                    }
                ],
                "Duration": "15000",
                "Score": 98,
                "Extra": "",
                "SpeakerResults": [],
                "LabelResults": [],
                "TravelResults": [],
                "SubLabel": "Crowd",
                "RecognitionResults": []
            },
            "OffsetTime": "120",
            "CreatedAt": null
        }
    ],
    "TryInSeconds": 0,
    "CreatedAt": null,
    "UpdatedAt": null,
    "ErrorType": "",
    "ErrorDescription": "",
    "Asrs": [],
    "Label": "",
```

```
    "SegmentCosUrlList": null
}
```

审核结束回调

```
{
    "TaskId": "w-video-ZSS_LFTB3lSvDHmP",
    "DataId": "test",
    "BizType": "default",
    "Name": "测试视频",
    "Status": "FINISH",
    "Type": "VIDEO",
    "Suggestion": "Block",
    "Labels": [
        {
            "Label": "Porn",
            "Suggestion": "Block",
            "Score": 99,
            "SubLabel": "PornSum"
        },
        {
            "Label": "Terror",
            "Suggestion": "Review",
            "Score": 99,
            "SubLabel": "Crowd"
        }
    ],
    "InputInfo": null,
    "MediaInfo": {
        "Codecs": "",
        "Duration": 144,
        "Width": 0,
        "Height": 0,
        "Thumbnail": ""
    },
    "AudioText": "xxxxxxxxxxxxxxxxxxx",
    "ImageSegments": [
        {
            "Result": {
```

```
            "HitFlag": 1,
            "Label": "Porn",
            "Suggestion": "Block",
            "Score": 99,
            "Results": [
                {
                    "Scene": "OCR",
                    "HitFlag": 1,
                    "Suggestion": "Block",
                    "Label": "Porn",
                    "SubLabel": "PornSum",
                    "Score": 99,
                    "Names": [],
                    "Text": "xxxxxxxxxxxxxxxxxxx",
                    "Details": [
                        {
                            "Name": "",
                            "Text": "xxxxxxxxxxxxxxxxxxx",
                            "Location": {
                                "X": 0,
                                "Y": 0,
                                "Width": 0,
                                "Height": 0,
                                "Rotate": 0
                            },
                            "Label": "Polity",
                            "LibId": "",
                            "LibName": "",
                            "Keywords": [
                                "xxx",
                                "xxx",
                                "xxx"
                            ],
                            "Suggestion": "Block",
                            "Score": 99,
                            "SubLabelCode": "PornSum",
                            "SubLabel": "PornSum"
                        }
                    ]
                }
```

```
                ],
                "Url": "https://cos.ap-
guangzhou.myqcloud.com/yunyingpingtai-xxx/_cms_segments/cms/vod/w-video-
ZSS_LFTB3lSvDHmP/screenshot_86_1696907076.jpg",
                "Extra": "{}",
                "SubLabel": "PornSum",
                "RecognitionResults": []
            },
            "OffsetTime": "86",
            "CreatedAt": "2023-10-10T11:04:37Z",
            "OffsetusTime": "86"
        }
    ],
    "AudioSegments": [
        {
            "Result": {
                "HitFlag": 1,
                "Url": "https://cos.ap-
guangzhou.myqcloud.com/yunyingpingtai-xxx/_cms_segments/cms/vod/w-video-
ZSS_LFTB3lSvDHmP/audio_60_1696907073.mp3",
                "Suggestion": "Block",
                "Label": "Polity",
                "Text": "xxxxxxxxxxxxxxxxxxx",
                "TextResults": [
                    {
                        "Label": "Terror",
                        "Score": 0,
                        "Keywords": [],
                        "Suggestion": "Pass",
                        "LibId": "",
                        "LibType": 0,
                        "LibName": "",
                        "SubLabel": ""
                    },
                    {
                        "Label": "Porn",
                        "Score": 2,
                        "Keywords": [],
                        "Suggestion": "Pass",
                        "LibId": "",
```

```
            "LibType": 0,
            "LibName": "",
            "SubLabel": ""
        },
        {
            "Label": "Polity",
            "Score": 99,
            "Keywords": [],
            "Suggestion": "Block",
            "LibId": "",
            "LibType": 0,
            "LibName": "",
            "SubLabel": "NationalInstitution"
        },
        {
            "Label": "Ad",
            "Score": 0,
            "Keywords": [],
            "Suggestion": "Pass",
            "LibId": "",
            "LibType": 0,
            "LibName": "",
            "SubLabel": ""
        },
        {
            "Label": "Abuse",
            "Score": 0,
            "Keywords": [],
            "Suggestion": "Pass",
            "LibId": "",
            "LibType": 0,
            "LibName": "",
            "SubLabel": ""
        },
        {
            "Label": "Illegal",
            "Score": 0,
            "Keywords": [],
            "Suggestion": "Pass",
            "LibId": "",
```

```
                "LibType": 0,
                "LibName": "",
                "SubLabel": ""
            },
            {
                "Label": "Spam",
                "Score": 0,
                "Keywords": [],
                "Suggestion": "Pass",
                "LibId": "",
                "LibType": 0,
                "LibName": "",
                "SubLabel": ""
            }
        ],
        "MoanResults": [
            {
                "Label": "Moan",
                "Score": 3,
                "StartTime": 0,
                "EndTime": 15,
                "Suggestion": "Pass",
                "SubLabel": ""
            }
        ],
        "LanguageResults": [
            {
                "Label": "yue",
                "Score": 90,
                "StartTime": 0,
                "EndTime": 15
            },
            {
                "Label": "dialect",
                "Score": 9,
                "StartTime": 0,
                "EndTime": 15
            }
        ],
        "Duration": "15000",
```

```
                "Score": 99,
                "Extra": "",
                "SpeakerResults": [],
                "LabelResults": [],
                "TravelResults": [],
                "SubLabel": "NationalInstitution",
                "RecognitionResults": []
            },
            "OffsetTime": "60",
            "CreatedAt": "2023-10-10T11:04:34Z"
        }
    ],
    "TryInSeconds": 0,
    "CreatedAt": "1970-01-01T00:00:00Z",
    "UpdatedAt": null,
    "ErrorType": "",
    "ErrorDescription": "",
    "Asrs": [
        {
            "text": "xxxxxxxxxxxxxxxxxxx",
            "CreatedAt": "2023-10-10T11:04:34Z"
        },
        {
            "text": "xxxxxxxxxxxxxxxxxxx",
            "CreatedAt": "2023-10-10T11:04:38Z"
        }
    ],
    "Label": "Porn",
    "SegmentCosUrlList": null
}
```


接入失败

若 Response 回参显示“Error”字段，则说明接入失败，以下为 signature 校验失败的示例：


```
{
    "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"
    }
}
```

其中 Error 中的 Code 表示错误码，Message 表示错误的具体信息，详情请参见 错误码 。

回调配置说明

最近更新时间：2026-09-08 17:18:33

回调准备工作

确定回调 URL

要开始设置回调服务，请首先确定您要接收回调的 URL。这是您的系统将接收来自我们服务器的 HTTP POST 请求的地址。确保您的 URL 是公共可访问的，并支持 HTTPS 协议。

配置回调参数

在您的系统中，您需要配置一些参数以接收和处理我们发送的 HTTP POST 请求。以下是一些重要的参数：

- 请求方法：HTTP POST。
- 请求头：Content-Type: application/json。
- 请求体：JSON 格式的数据，包含有关回调结果的详细信息，详情信息请参考接收回调结果。

您需要根据您的系统需求调整这些参数。请确保您的系统能够解析和处理 JSON 格式的请求体。

测试回调服务

在实际使用回调服务之前，请确保您已经成功地设置了回调 URL 和参数，并且您的系统已经可以接收和处理来自我们服务器的 HTTP POST 请求。您可以使用一些在线工具来模拟 HTTP POST 请求，以确保您的回调服务正常工作。

处理回调事件

当我们的服务器发送 HTTP POST 请求到您的回调 URL 时，您的系统需要解析请求体中的 JSON 数据，并根据事件状态 Status 执行相应的操作。以下是一些常见的事件类型：

- FINISH：任务审核成功。
- RUNNING：任务还在执行中。
- ERROR：任务执行失败，可参考回调中的失败类型（ErrorType）和具体原因（ErrorDescription）字段来确认失败原因。

请根据您的系统需求来处理不同的事件类型。请确保响应的 HTTP 状态码为200 OK。

接收回调结果

请查阅 接入后验证 以获取回调示例，字段释义详见 查询任务详情 接口文档。

> ⓘ **注意：**
> 在点播场景中，默认情况下只有当审核结束（Finish）时才会触发回调。而在直播场景中，默认情况下只有在检测到违规和审核结束时才会触发回调。

RUNNING 输入参数

任务审核中，该状态会对单个图片或音频片段的检测结果进行回调。

FINISH 输入参数

任务审核结束，该状态下会返回前30个图片 / 音频违规片段结果，完整结果可通过 RUNNING 片段回调去拼接，或者调用查询任务详情接口来获取。

验签流程

为了验证接收到的回调数据是否来自合法的发送方并且在传输过程中未被篡改，用户在创建任务时可以选择向我们传递 seed 值，如果不需要验签则可以不用传递。在回调时，我们会加入签名参数以确保数据的安全性。我们将在返回的 HTTP 头部中添加 `X-Signature` 字段，其值将为 `seed` + `body` 的 sha256 编码和 Hex 字符串。

ⓘ **例如：用户 CallbackUrl 是：** `http://example.com` **，Seed 是：** `dedb6dcc1cb7c63fde8fa5abfd57` **，我们返回的回调的数据是：**
{"TaskId": "task-video-X0zpcRUMzVidxj20","DataId":"test","Suggestion": "Block"}
则，审核完成后，我们会在调用 `http://example.com` **的时候，在 HTTP 头部 传入** `X-Signature` **的值为：** `74f0ae6d1f1e4eb1ffe4162da480a812f8a4dc19fe5a52bacbcd2c862d3edcfd`

```python
#!/usr/bin/env python3
# -*- encoding: utf-8 -*-
# 验签示例

import hashlib

seed = "dedb6dcc1cb7c63fde8fa5abfd57"
body = '{"TaskId": "task-video-
X0zpcRUMzVidxj20","DataId":"test","Suggestion": "Block"}'
# 将 seed + body 转为 bytes 类型，并计算 SHA256 编码
sha256 = hashlib.sha256((seed + body).encode('utf-8')).hexdigest()
print(sha256)
```

Running Environment

Operating System: Ubuntu 24.04.3 LTS / x86_64

Runtime Version: Python 3.11.1

回调 URL 续签

由于天御只具有临时访问用户 COS 桶的权限，临时权限最长只有 12 小时，因此，回调的 URL 有效期最长为 12 小时。建议在接收到回调时立即对 URL 进行签名更新和替换，以获取更长时效的链接。COS 目前支持的各语言 SDK 都提供了生成签名链接的功能，但获取长期时效链接需要使用 永久密钥 。以下提供了 Java、Python、Go 对应 SDK 调用的示例。

说明：

回调 URL 格式：`https://<bucket>.cos.region.myqcloud.com/file_path`

- 回调 URL 示例：`https://tianyu-live-video-content-moderation-xxxx.cos.ap-guangzhou.myqcloud.com/segment-/audio/w-live_video-YrvumJyVBc3KM1DF/1656483481.mp3`
- 续签后 URL 示例：`https://tianyu-live-video-content-moderation-xxxx.cos.ap-guangzhou.myqcloud.com/segment-/audio/w-live_video-YrvumJyVBc3KM1DF/1656483481.mp3?q-sign-algorithm=sha1&q-ak=AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&q-sign-time=1754477817;1755082617&q-key-time=1754477817;1755082617&q-header-list=host&q-url-param-list=&q-signature=1ea700f9e6431787d22c5ecc6a6b4a1347909d91`

Python

```python
#!/usr/bin/env python3
# -*- coding:utf-8 -*-

from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client

# 密钥
secret_id = "" # 点击永久密钥获取
secret_key = "" # 点击永久密钥获取
region = "" # 默认 ap-guangzhou


def get_cos_presigned_url(url):
    """
    获取.....
    :param url:
    :return: presigned_url
    """
    # 获取客户端对象
```

```
    config = CosConfig(Region=region, SecretId=secret_id,
SecretKey=secret_key)
    client = CosS3Client(config)
    # 根据key获取下载链接
    # 获取签名的cos桶
    bucket_name = "tianyu-live-video-content-moderation-xxxx"
    # print(bucket_name)
    # 获取签名的 COS 路径，请确保路径不被编码
    key = "segment-/audio/w-live_video-YrvumJyVBc3KM1DF/1656483481.mp3"
    # print(key)
    # 使用永久密钥时，expired_sec 签名过期时间可以随意设置
    expired_sec = 60 * 60 * 24 * 7
    download_response = client.get_presigned_download_url(
        Bucket=bucket_name, Key=key, Expired=expired_sec)
    return download_response


if __name__ == '__main__':
    # presigned_url既为获得新的签名的URL
    presigned_url = get_cos_presigned_url(
        "url")
    print(presigned_url)
```

Java

```
import com.qcloud.cos.COSClient;
import com.qcloud.cos.ClientConfig;
import com.qcloud.cos.auth.BasicCOSCredentials;
import com.qcloud.cos.auth.COSCredentials;
import com.qcloud.cos.http.HttpMethodName;
import com.qcloud.cos.http.HttpProtocol;
import com.qcloud.cos.region.Region;
import java.net.URL;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

public class TencentCosClient{
    public static void main(String [] args){
```

```
        // 调用 COS 接口之前必须保证本进程存在一个 COSClient 实例，如果没有则创建
        // 详细代码参见本页：简单操作 -> 创建 COSClient
        COSClient cosClient = createCOSClient();

        // 获取签名的cos桶
        String bucketName = "tianyu-live-video-content-moderation-xxxx";
        // 获取签名的 COS 路径，请确保路径不被编码
        String key = "segment-/audio/w-live_video-
YrvumJyVBc3KM1DF/1656483481.mp3";

        // 设置签名过期时间(可选)， 若未进行设置则默认使用 ClientConfig 中的签名过
期时间(1小时)
        // 这里设置签名在半个小时后过期
        Date expirationDate = new Date(System.currentTimeMillis() + 30 *
60 * 1000);

        // 填写本次请求的参数，需与实际请求相同，能够防止用户篡改此签名的 HTTP 请求
的参数
        Map<String, String> params = new HashMap<String, String>();
        params.put("param1", "value1");

        // 填写本次请求的头部，需与实际请求相同，能够防止用户篡改此签名的 HTTP 请求
的头部
        Map<String, String> headers = new HashMap<String, String>();
        headers.put("header1", "value1");

        // 请求的 HTTP 方法，上传请求用 PUT，下载请求用 GET，删除请求用 DELETE
        HttpMethodName method = HttpMethodName.GET;

        URL url = cosClient.generatePresignedUrl(bucketName, key,
expirationDate, method, headers, params);
        System.out.println(url.toString());

        // 确认本进程不再使用 cosClient 实例之后，关闭之
        cosClient.shutdown();
    }
    // 创建 COSClient 实例，这个实例用来后续调用请求
    public static COSClient createCOSClient() {
        // 设置用户身份信息。
```

```
        // SECRETID 和 SECRETKEY 请登录访问管理控制台
https://console.cloud.tencent.com/cam/capi 进行查看和管理
        String secretId = "";
        String secretKey = "";
        COSCredentials cred = new BasicCOSCredentials(secretId,
secretKey);

        // ClientConfig 中包含了后续请求 COS 的客户端设置:
        ClientConfig clientConfig = new ClientConfig();

        // 设置 bucket 的地域
        // COS_REGION 请参照
https://cloud.tencent.com/document/product/436/6224
        clientConfig.setRegion(new Region("ap-guangzhou"));

        // 设置请求协议, http 或者 https
        // 5.6.53 及更低的版本，建议设置使用 https 协议
        // 5.6.54 及更高版本，默认使用了 https
        clientConfig.setHttpProtocol(HttpProtocol.https);

        // 以下的设置，是可选的:
        // 设置 socket 读取超时，默认 30s
        clientConfig.setSocketTimeout(30*1000);
        // 设置建立连接超时，默认 30s
        clientConfig.setConnectionTimeout(30*1000);

        // 如果需要的话，设置 http 代理，ip 以及 port
        clientConfig.setHttpProxyIp("httpProxyIp");
        clientConfig.setHttpProxyPort(80);

        // 生成 cos 客户端。
        return new COSClient(cred, clientConfig);
    }
}
```

Go

```
package main
```

```
import (
    "context"
    "fmt"
    "net/http"
    "net/url"
    "time"

    "github.com/tencentyun/cos-go-sdk-v5"
)

// 通过tag的方式，用户可以将请求参数或者请求头部放进签名中。
type URLToken struct {
    SessionToken string `url:"x-cos-security-token,omitempty" header:"-
"`
}

func GetCosUrl() {
    // 替换成您的密钥
    tak := ""
    tsk := ""

    tmpURL := "https://tianyu-live-video-content-moderation-
xxxx/segment-/audio/w-live_video-YrvumJyVBc3KM1DF/1656483481.mp3?X-Amz-
Algorithmxxxx"

    parse, err := url.Parse(tmpURL)
    if err != nil {
        fmt.Println(err)
        return
    }
    // 获取签名的cos桶 tianyu-live-video-content-moderation-xxxx
    bucket_name := parse.Scheme + "://" + parse.Host
    // 获取签名的 COS 路径，请确保路径不被编码 /segment-/audio/w-live_video-
YrvumJyVBc3KM1DF/1656483481.mp3
    key := parse.Path

    u, _ := url.Parse(bucket_name)
    b := &cos.BaseURL{BucketURL: u}
    c := cos.NewClient(b, &http.Client{})
```

```
    name := key
    ctx := context.Background()

    // 通过 tag 设置 x-cos-security-token
    // Get presigned
    presignedURL, err := c.Object.GetPresignedURL(ctx, http.MethodGet,
name, tak, tsk, 100*time.Hour, nil)
    if err != nil {
        fmt.Printf("Error: %v\n", err)
        return
    }
    // Get object by presinged url
    resp, err := http.Get(presignedURL.String())
    if err != nil {
        fmt.Printf("Error: %v\n", err)
    }
    defer resp.Body.Close()
    fmt.Println(presignedURL.String())
    fmt.Printf("resp:%v\n", resp)
}
```

回调异常常见问题

未收到回调结果一般是什么原因?

1. 任务还在 Pending 中，可以查询任务详情接口看 Status 状态。任务审核逻辑是新增任务优先审核，旧任务往后排队。
2. 当配置为最终结果回调时，只有在审核任务 Finish 结束后才会触发回调。您可以通过详情接口查看 Status 任务状态，也可以通过控制台修改回调逻辑，以支持片段和最终结果的回调。
3. 用户侧未按照正确的响应码响应200，导致审核回调失败，可以使用类似 Postman 或 curl 等工具发送请求，并查看响应代码是否为200。
4. 回调等待响应头时超时，回调超时时间超过5s 会报错"context deadline exceeded (Client.Timeout exceeded while awaiting headers)" ，该情况需要提供 taskid 给到产品侧确认。
    - 检查回调服务是否正常运行，并确保其能够及时响应请求。
    - 检查网络连接是否稳定，并尝试使用更快的网络连接。
    - 如果请求被防火墙或其他网络安全设备拦截，可以尝试修改其配置，以允许请求通过。

建议客户侧可以加上调试和监控功能系统。可以使用日志记录来记录请求和响应数据，以便进行故障排除。您还可以使用监控工具来监视您的回调服务的性能和可用性。

回调 URL 为什么无法打开?

如果用户的接收程序对回调的内容进行了编码，会导致链接无法正确打开。

错误码

最近更新时间：2026-09-08 17:18:30

公共错误码

错误码	说明
ActionOffline	接口已下线，停止服务
AuthFailure.InvalidAuthorization	请求头部的 Authorization 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
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.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

InternalError.ErrTextTimeOut	请求超时
InvalidParameter.ErrAction	错误的 action
InvalidParameter.ErrTextContentLen	请求的文本长度过长
InvalidParameter.ErrTextContentType	文本类型错误，需要 base64 的文本
InvalidParameterValue.ErrTextContentLen	请求的文本长度超过限制
InvalidParameterValue.ErrTextContentType	请求的文本格式错误，需要 base64 编码格式的文本
UnauthorizedOperation.Unauthorized	未获取到接口授权