视频内容安全

实践教程

腾讯云

文档目录

实践教程
接入指引
接入准备

最近更新时间：2025-02-05 16:07:53

本文档将介绍如何接入和使用视频内容安全（Video Moderation System，VM）。

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 VM 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。

API 接入指引

最近更新时间：2024-11-08 10:45:22

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。

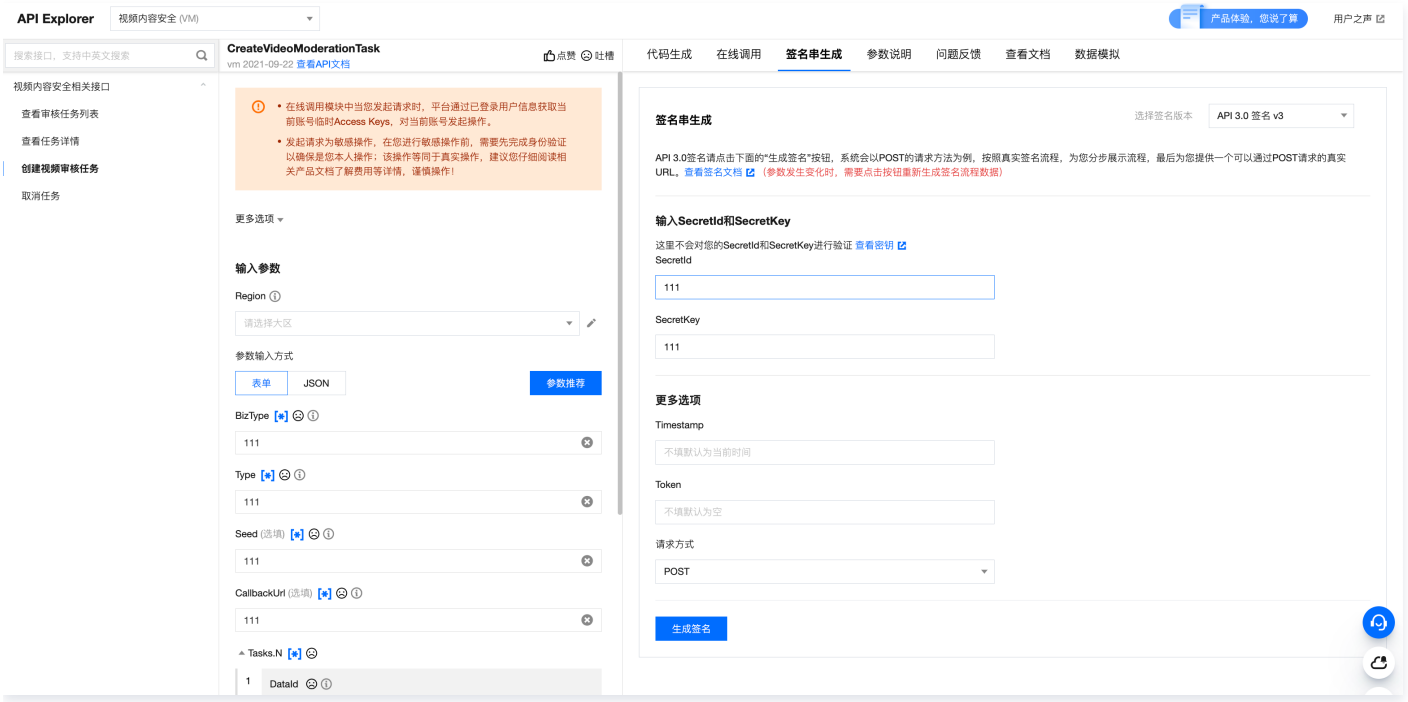
公共参数

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

```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.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：2025-02-05 16:07:53

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

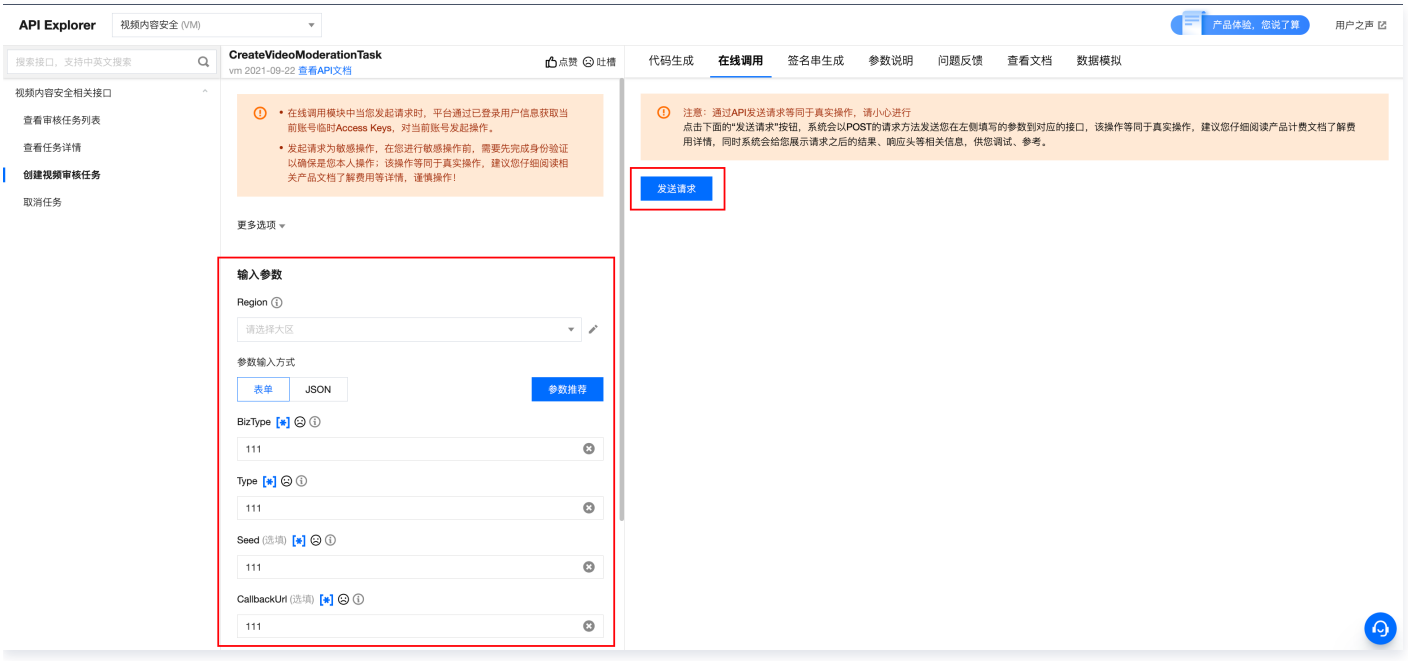
参考 视频内容安全 接口文档，了解需要使用的参数。

SDK 使用说明

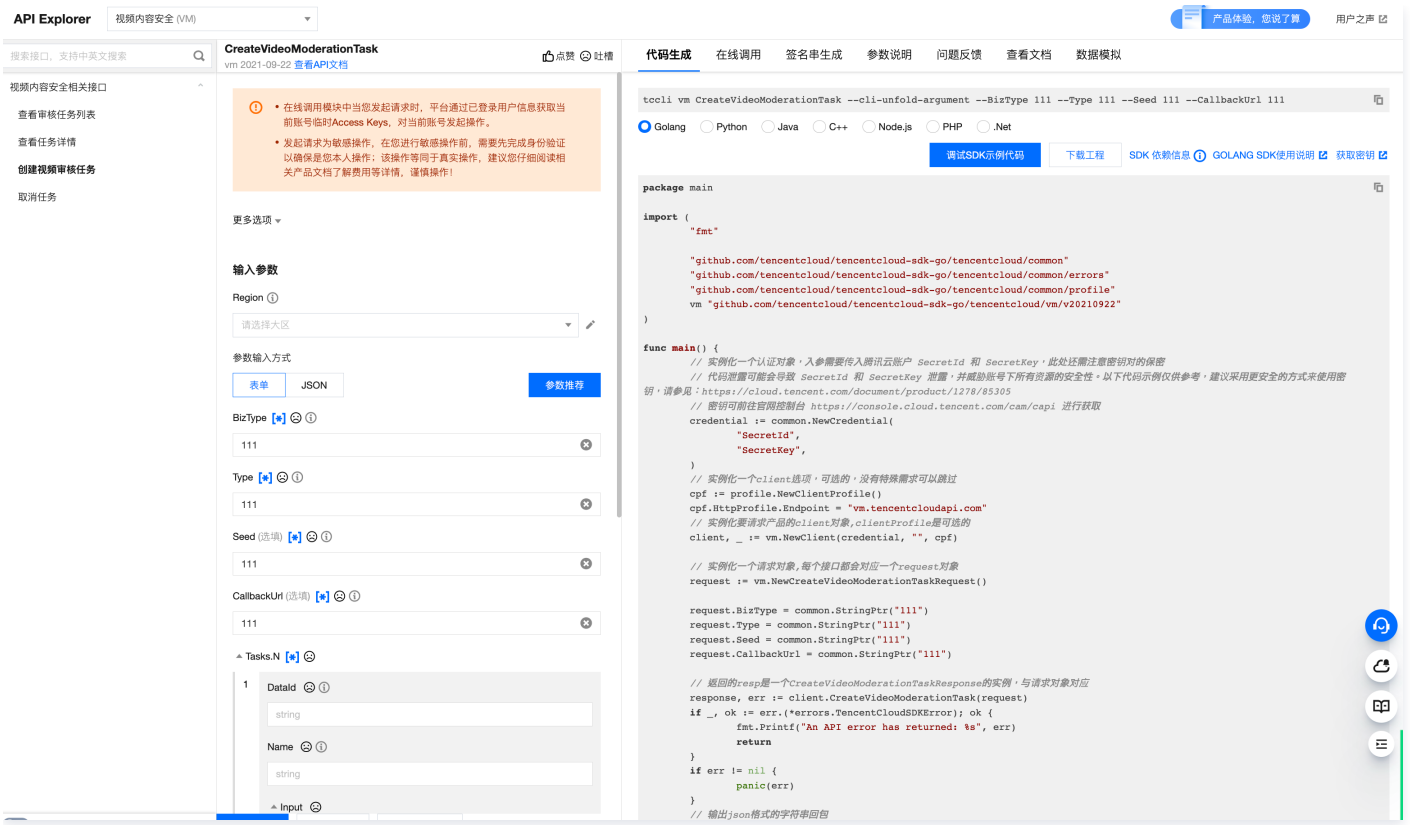
云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，均已开源，能更方便的调用 API，支持 Python、Java、PHP、Go、NodeJS、.NET、C++。
各语言 SDK 使用说明可参考 文档，介绍依赖版本以及安装过程等。

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：2025-02-05 16:07:53

当您通过上述步骤完成 API 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": "xxxxxxxxxxxxxxxxxx",
                        "Details": [
                            {
                                "Name": "",
                                "Text": "xxxxxxxxxxxxxxxxxx",
                                "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": "xxxxxxxxxxxxxxxxxxxx",
                "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：2024-12-10 17:15:01

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。

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 FAQ

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。

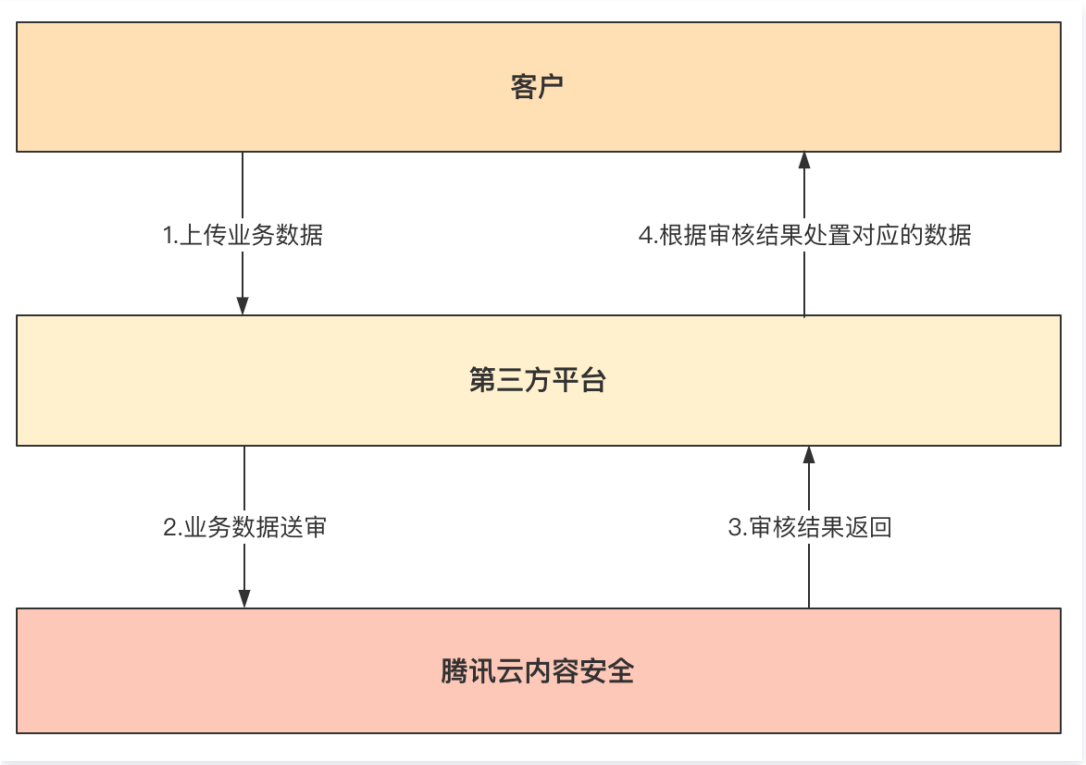
业务实践
第三方送审视频审核

最近更新时间：2024-09-05 20:49:41

概述

内容安全分为 文本内容安全 、图片内容安全 、音频内容安全 、视频内容安全 四个产品。内容安全支持用户通过第三方平台接入内容安全服务，如您在第三方平台已有相应的业务数据信息，您可以选择由第三方平台协助您完成数据快速送审工作，同时内容安全产品团队会将审核结果返回至第三方平台，以便第三方平台进行相应的配置，相关审核数据可以在内容安全控制台查看。

> **说明**
> 内容安全分为文本、图片、音频、视频四个场景，您可在 步骤2：配置场景 中，根据需要审核的类型，配置对应场景。



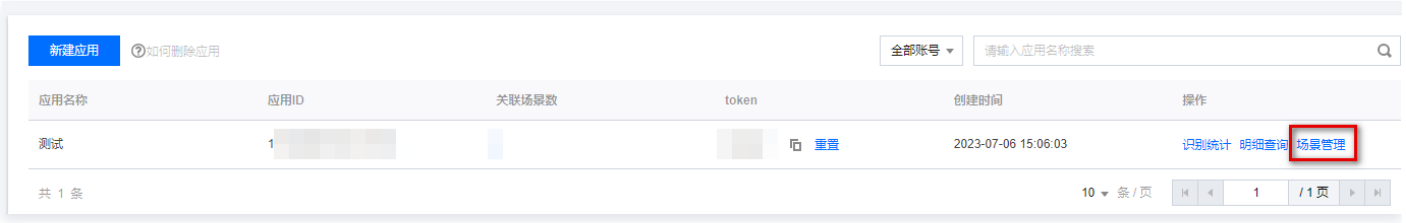
步骤1：新建应用

1. 登录 内容安全控制台 ，在**概览页**单击**应用管理** > **新增应用**，新增需要配置视频内容安全管理的应用。

2. 在应用管理页面，单击**新增应用**，输入应用名称，单击**保存**完成应用创建。

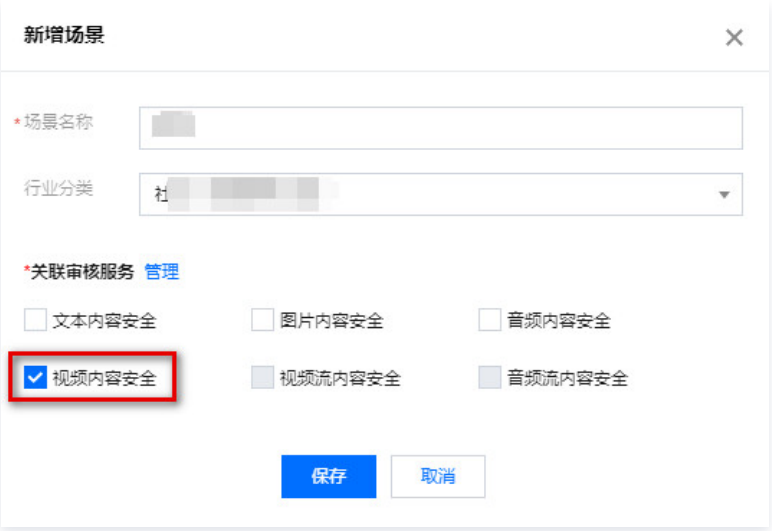
步骤2：配置场景

1. 在应用管理页面，选择已创建的应用，单击**场景管理**。



2. 在场景管理页面，单击**新建场景**，输入场景名称、选择行业分类、选择**视频内容安全**，单击**保存**即可完成场景创建。

> **说明：**
> 在审核服务管理中开启的审核服务可勾选，未开启的审核服务不可选。若想修改审核服务管理中的选项可单击**管理**修改。



参数名称	描述
场景名称	场景的文字描述，可使用中文、英文、数字及下划线组合，长度不超过20个字符。
行业分类	策略涉及的行业场景分类。
关联审核服务	选择场景需要关联的审核服务。

3. 场景创建完成，开始配置场景策略，单击**配置** > **视频审核内容安全**。

4. 在策略详情页面会显示当前系统默认策略，如需更改，单击**编辑**，即可修改当前策略。

参数名称	描述
识别策略配置	根据业务需求，选择是否需要识别不同类型的识别内容。
截帧策略配置	• 审核内容配置：可选全部审核、仅审核视频画面、仅审核音频。 • 图片截帧间隔，设置图片截帧的时间长度，可自行填写设置，默认为1s。 • 音频切片时长，设置音频切片的时间长度，可选15s、30s、60s，也可自行填写设置，默认为30s。 • 审核结果回调配置： ○ 违规回调：仅返回违规审核结果。 ○ 全量回调：返回全部审核结果。

关联自定义库	选择是否需要使用自定义词库用于内容识别，如暂无自定义库，可跳过，或保存当前策略后前往 快速入门-步骤五：配置自定义词库 。

5. 完成后单击**确认**，完成策略配置

步骤3：获取 Token 和场景送审 URL

说明

- Token：用于调用审核服务时的身份验证。
- 场景送审 URL：内容安全审核服务的调用地址。

1. 在应用管理页面，选择目标应用，单击复制送审 Token，**并跳转回第三方平台进行配置**。

说明

如您重置 Token，则在其他第三方渠道配置的原 Token 将会失效，需要您重新在第三方添加最新的 Token。

2. 在应用管理页面，选择已创建的应用，单击**场景管理**。

3. 在场景管理页面，选择目标场景，单击**更多 > 复制送审 URL**，并跳转回第三方平台进行配置。

步骤4：跳转回第三方平台完成数据送审

在第三方平台配置内容安全审核服务的 Token 和场景送审 URL 后，**请根据第三方平台后续的指引，完成数据送审**。如有疑问，请 联系我们 。

说明

用户可登录 内容安全控制台 ，查看送审的明细数据、统计数据，并进行策略配置等操作。

即时通信 IM 云端审核

最近更新时间：2024-12-10 17:00:53

即时通信 IM 云端审核功能，是天御内容安全产品，在云端进行内容智能识别和拦截的服务实践。

- 支持对文本、图片内容同步审核，对音频、视频内容异步审核。
- 支持分别针对单聊、群聊、资料三大场景中产生的不同类型内容进行策略配置。
- 支持配置特定终端用户产生/接收的消息不送审，支持对返回审核结果进行配置。
- 支持查看审核用量抵扣明细、审核结果明细、审核识别统计等数据。
- 能够精准识别文本中出现可能令人反感、不安全或不适宜的内容，有效降低内容违规风险与有害信息识别成本。

步骤1：购买 IM 云端审核资源包

请前往 即时通信 IM 购买页，购买云端审核资源包。体验版、专业版、旗舰版应用均可购买，购买完成后即刻生效。

> **注意：**
> 开通服务后，默认为您开启图文审核场景，并配置默认审核策略（可识敏感事件、敏感人物、色情、辱骂、违禁等不安全内容），您可通过单击对应模块的**编辑**按钮，分别更改审核场景和审核策略。

步骤2：开启 IM 云端审核功能开关

1. 登录 即时通信 IM 控制台，在左侧导航栏中，选择**内容审核** > **云端审核**。
2. 在云端审核页面，单击页面右上角的，开启云端审核服务。

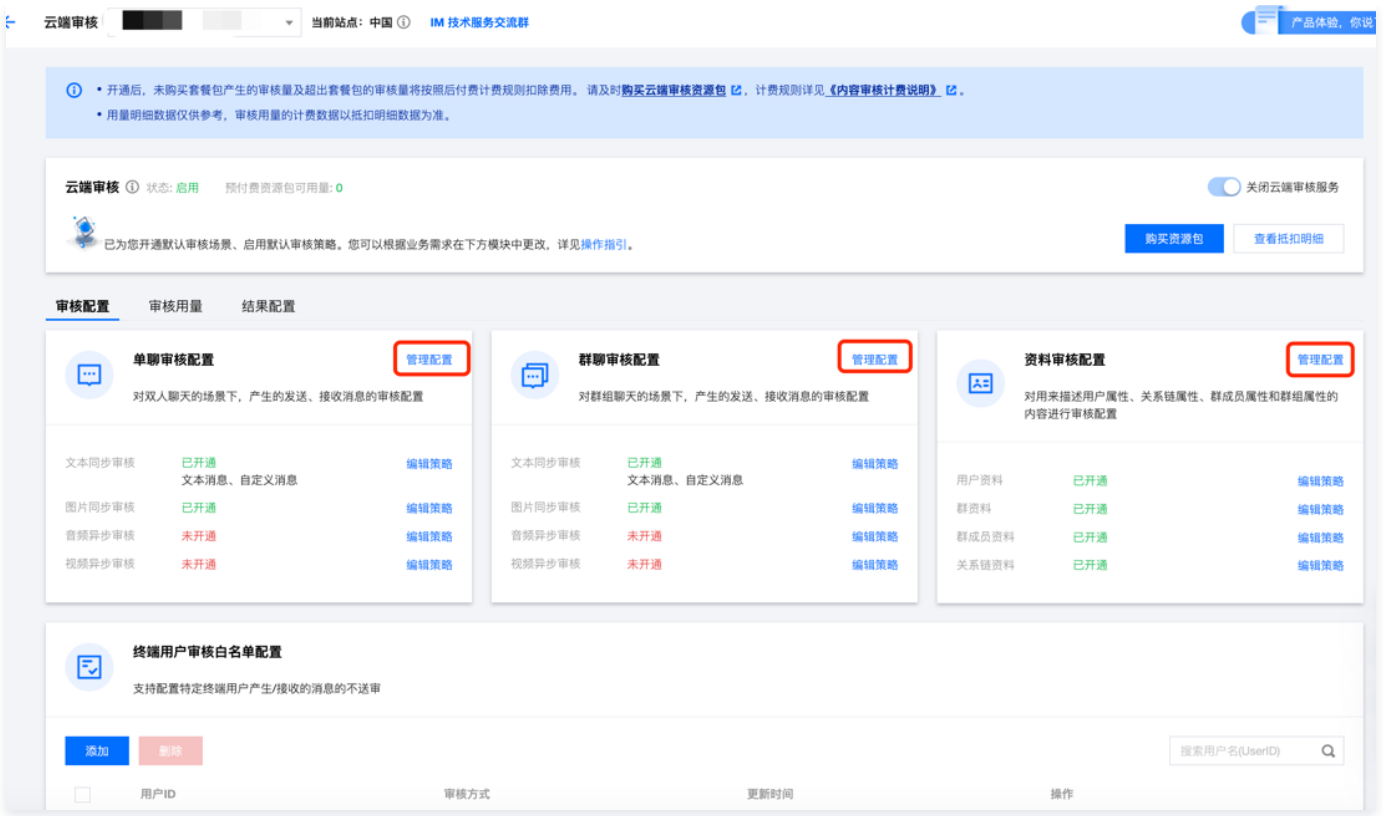
> **注意**
> 如果您未提前购买云端审核资源包，在控制台直接开通云端审核服务后，**产生的审核量将按照后付费计费规则扣除费用**，请及时 购买云端审核资源包，计费规则详见 计费说明。

步骤3：自定义审核配置

审核场景配置

打开云端审核服务开关后，默认为您开启了单聊的文本和图片审核、群聊的文本和图片审核、用户资料/群资料/群成员资料/关系链资料的文本审核。

1. 如果需要开启音频/视频审核，或者关闭图文审核开关，请单击对应场景模块右上角的**管理配置**。



2. 在弹窗中，勾选或取消勾选对应审核项，单击**确定**即可。

> **注意：**
> 单聊、群聊场景下，将音视/视频异步审核开启后，所有音频/视频都将正常送审并计费，但仅对存储在腾讯云即时通信 IM 的内容生效审核结果。如有自定义存储的音频/视频内容，请谨慎开启！如需审核自定义存储的音频/视频内容，请通过监听音频/视频审核结果的方式，进行拦截或撤回操作，详情请参见 返回审核结果 配置。

审核策略配置

每个审核场景及内容类型均已配置默认审核策略，默认审核策略包括敏感事件、敏感人物、色情、辱骂、违禁等风险的识别。如果需要编辑审核策略，请单击对应场景及内容类型右侧的**编辑策略**。

文本/图片策略配置

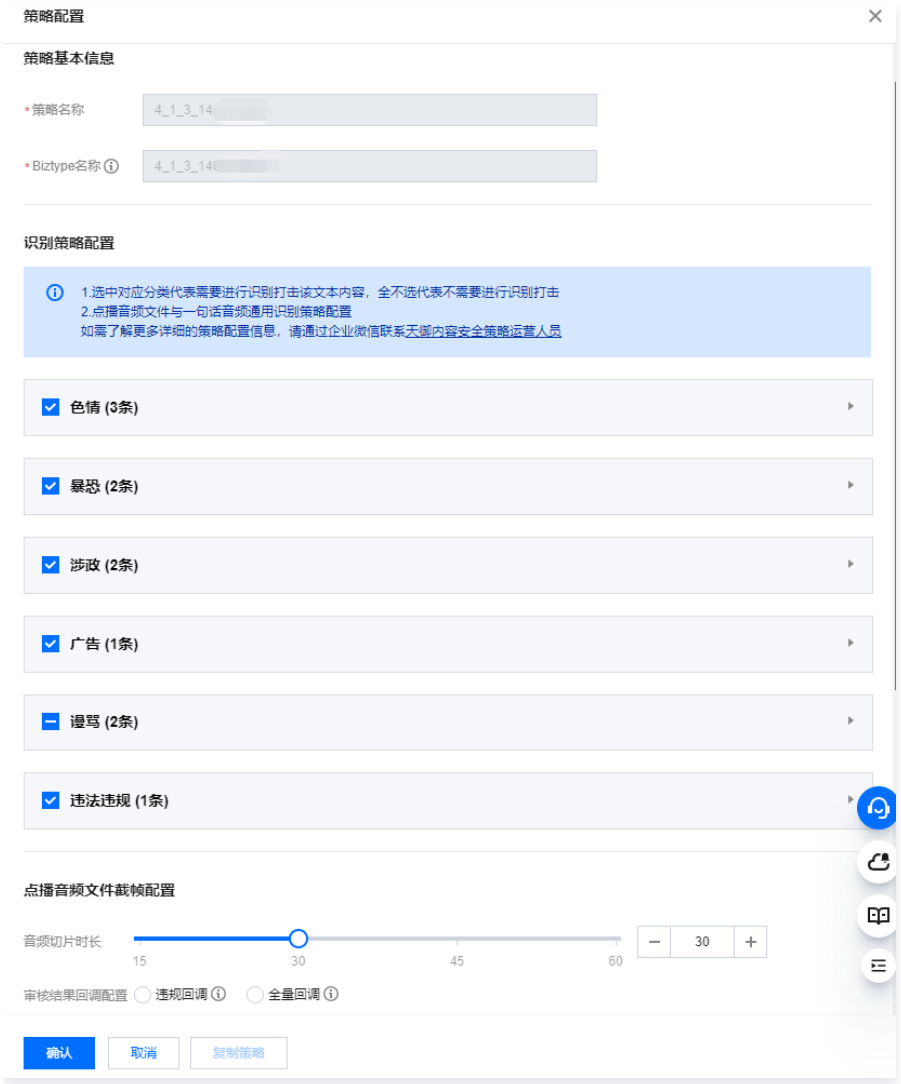
单击编辑，修改通用识别策略配置与自定义库信息，单击确认保存配置。

- **通用识别策略配置：勾选需要识别的风险类型。**
- **自定义库信息：勾选需要关联的自定义词库（最多可支持设置20个自定义词库）。**

音频策略配置

单击**编辑**，修改通用识别策略配置、截帧策略配置与自定义库信息，单击**确认**保存配置。

- 通用识别策略配置：勾选需要识别的风险类型。
- 截帧策略配置：音频切片时长配置，支持滑动设定或填写整数，范围[1,60]。
- 自定义库信息：勾选需要关联的自定义词库（最多可支持设置20个自定义词库）。



视频策略配置

单击**编辑**，修改通用识别策略配置、截帧策略配置与自定义库信息，单击**确认**保存配置。

- 通用识别策略配置：勾选需要识别的风险类型，包括对图片和音频的识别。
- 截帧策略配置：审核内容配置，支持仅审核视频画面、仅审核音频和全部审核；图片截帧间隔，支持滑动设定或填写整数，范围[1,60]；音频切片时长配置，支持滑动设定或填写整数，范围[1,60]。
- 自定义库信息：勾选需要关联的自定义词库（最多可支持设置20个自定义词库）。

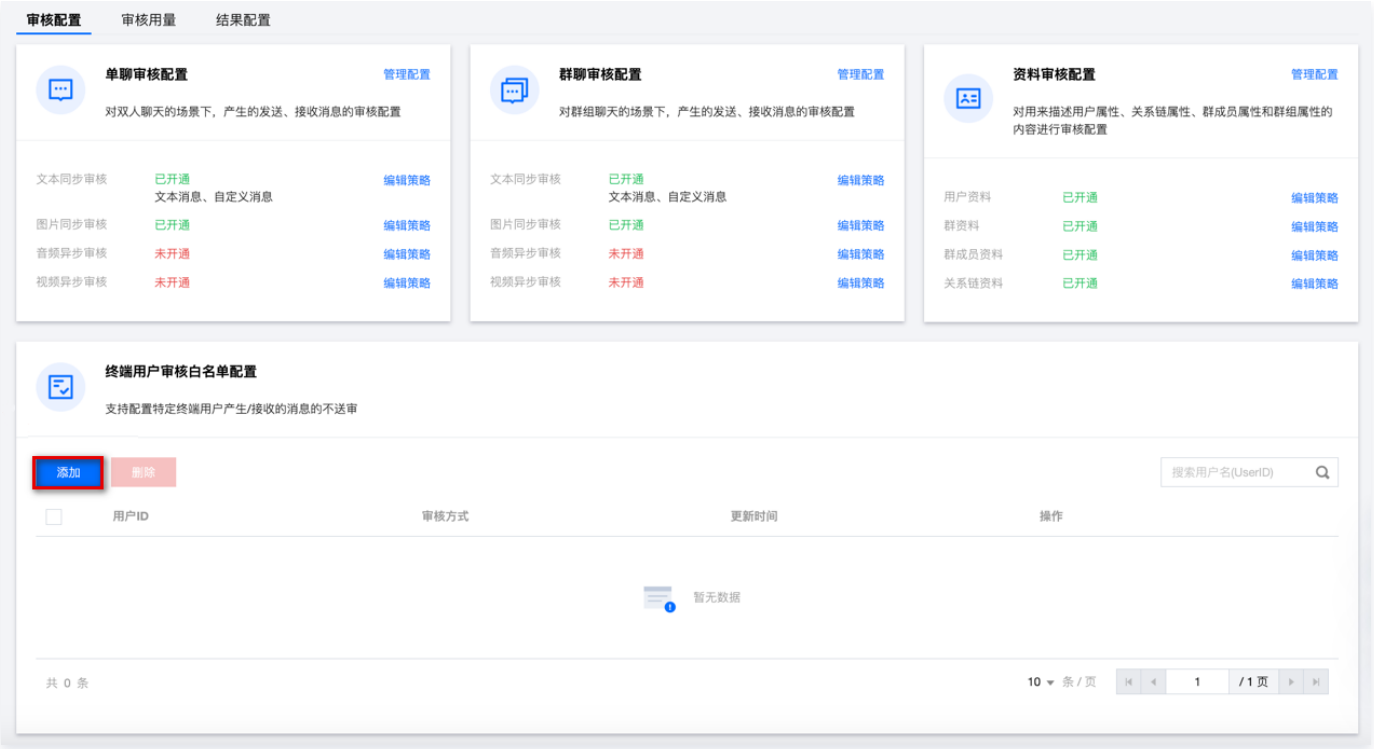
复制策略

图文音视的编辑策略时，单击**复制策略**，可以将本策略复制到下拉选择的其他场景，单击**确定**。

终端用户审核白名单配置

如果您不需要审核特定终端用户发送/接收的消息内容，您可以通过终端用户审核白名单模块进行配置。具体操作如下：

1. 在云端审核页面的终端用户审核白名单配置模块，单击**添加**。



2. 在弹窗中，选择审核方式，输入终端用户 ID，单击**确定**。

3. 在终端用户审核白名单配置模块，支持对已添加的终端用户 ID 编辑不送审方式、删除/批量删除添加白名单记录。

步骤4：审核结果配置

返回审核结果

审核结果可以通过配置基础回调的方式，将结果转发给 App 后台。具体操作如下：

1. 返回审核结果开关默认关闭，如果需要打开，单击审核结果配置模块右上角的**编辑**，打开返回审核结果开关，单击**确定**。

2. 内容回调结果将发送至您在基础配置中配置的 URL 地址，如果您在开启返回审核结果开关前未配置基础回调 URL，请在打开开关时弹出的弹窗中填写基础回调 URL，详情请参见 配置指引文档 。

3. 开关开启后，内容回调的结果将通过基础回调抄送到 App 后台，支持拦截、所有结果的配置选项。

> **注意**
> App 后台在收到回调请求之后，务必校验请求 URL 中的参数 SDKAppID 是否是自己的 SDKAppID。

下发拦截错误码

针对图文同步审核，默认下发拦截错误码，即消息发送方会收到消息被拦截的错误码提示。
如果需要发送方对被消息拦截无感知，即发送方不需要收到消息下发失败的提示，可通过关闭“下发拦截错误码”开关实现。具体操作如下：
在审核结果配置模块，单击右上角的**编辑**，关闭“下发拦截错误码”开关，单击**确定**。

注意

为方便您回溯消息未成功发送的原因，您需要在关闭“下发拦截错误码”开关前，开启 返回审核结果 开关，接收审核结果。

步骤5：查看审核明细和抵扣明细

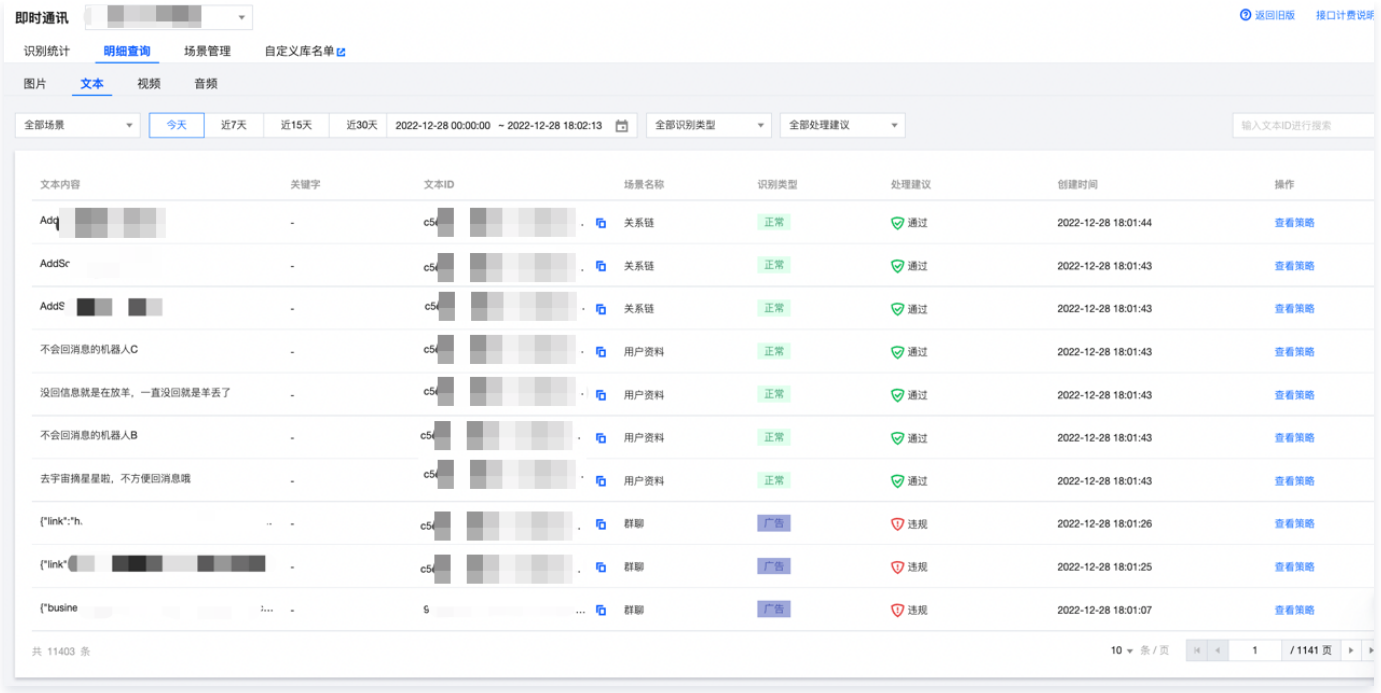
查看审核明细

在云端审核页面的审核用量中，可查看今日审核用量数据。

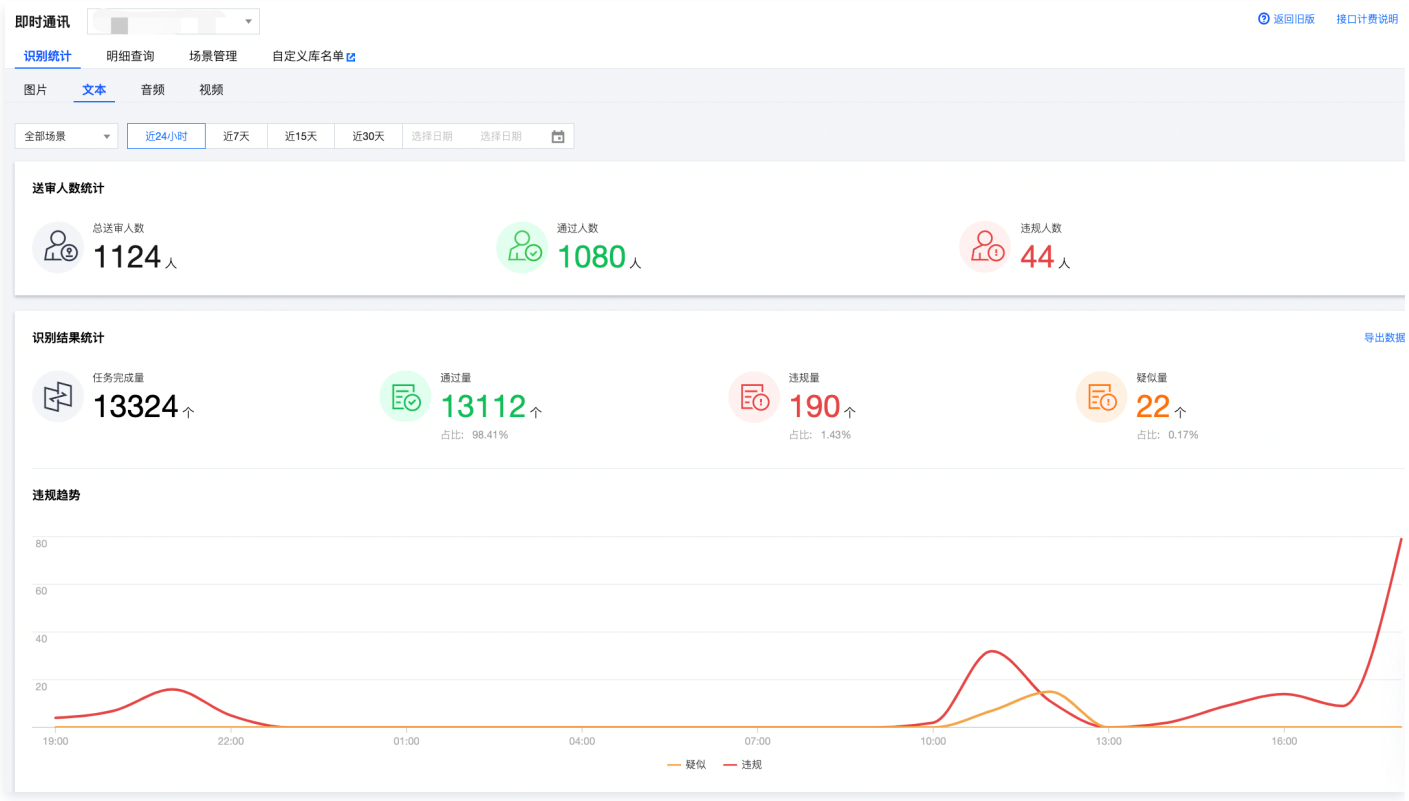
注意

用量明细数据仅供参考，审核用量的计费数据以抵扣明细数据为准。

- 单击模块右上角的**查看审核明细**，可查看历史审核的明细数据。

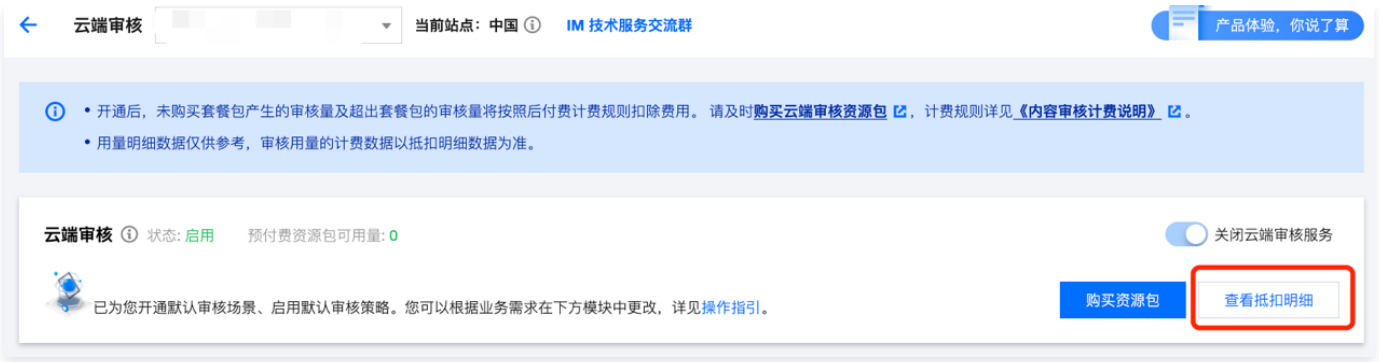


- 单击模块右上角的**查看识别统计**，可查看历史审核的统计数据。

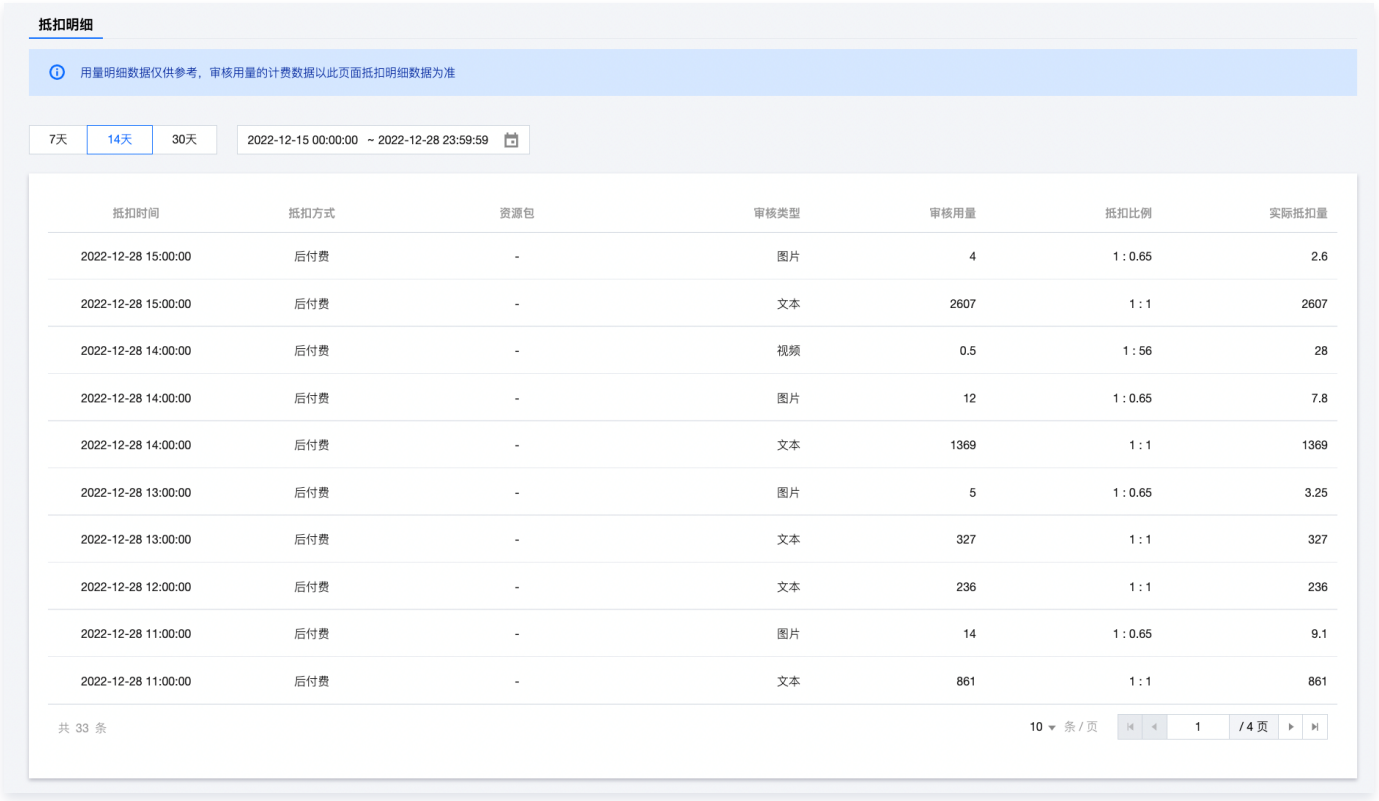


查看抵扣明细

1. 在云端审核页面，单击**查看抵扣明细**。



2. 在抵扣明细页面，支持筛选、查看特定日期内审核产生用量的具体抵扣数据（每小时更新）。



抵扣时间	抵扣方式	资源包	审核类型	审核用量	抵扣比例	实际抵扣量
2022-12-28 15:00:00	后付费	-	图片	4	1 : 0.65	2.6
2022-12-28 15:00:00	后付费	-	文本	2607	1 : 1	2607
2022-12-28 14:00:00	后付费	-	视频	0.5	1 : 56	28
2022-12-28 14:00:00	后付费	-	图片	12	1 : 0.65	7.8
2022-12-28 14:00:00	后付费	-	文本	1369	1 : 1	1369
2022-12-28 13:00:00	后付费	-	图片	5	1 : 0.65	3.25
2022-12-28 13:00:00	后付费	-	文本	327	1 : 1	327
2022-12-28 12:00:00	后付费	-	文本	236	1 : 1	236
2022-12-28 11:00:00	后付费	-	图片	14	1 : 0.65	9.1
2022-12-28 11:00:00	后付费	-	文本	861	1 : 1	861

共 33 条　10 条 / 页　1 / 4 页

3. 如果您已购买预付费资源包，可在资源包管理页面查看资源包可用量、有效期等信息。

直播流内容审核

最近更新时间：2024-09-05 20:49:41

对于视频流内容，在 开通天御视频内容安全服务 后，可直接调用**视频安全服务**接口，对视频流内容（如直播间、视频会议等）进行批量识别。

> **说明：**
> - 请在调用前确保目前账号**至少拥有视频内容安全服务的访问权限**，有关权限配置的相关信息，敬请参阅 CAM 授权指引 文档。
> - 若无法访问服务，则请开通服务/检查计费信息（主账号），或向管理员或主账号申请相应权限（子账号/协作者账号）。

步骤一：配置策略（可选）

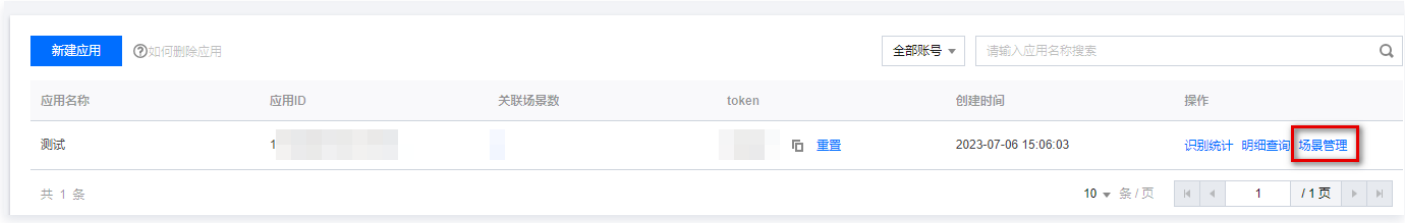
建议您使用配置任务策略，可根据业务需求配置识别策略，用于个性化服务体验。

> **说明**
> - 腾讯云内容安全服务已预设有默认策略，如使用默认策略，可略过此步骤。
> - 默认策略为天御多行业模型沉淀的策略配置，适用于大部分的内容安全需求。

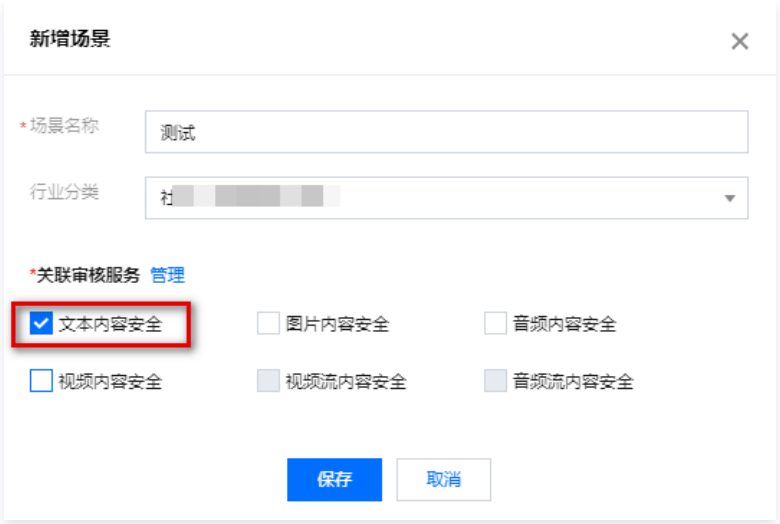
1. 登录 内容安全控制台，在**概览页**单击**应用管理 > 新增应用**，新增需要配置视频内容安全管理的应用。

2. 在应用管理页面，单击**新增应用**，输入应用名称，单击**保存**完成应用创建。

3. 在应用管理页面，选择刚创建的应用，单击**场景管理**。



4. 在场景管理页面，单击**新建场景**，输入场景名称、选择行业分类、选择文本内容安全，单击**保存**即可完成场景创建。



参数名称	描述
场景名称	场景的文字描述，可使用中文、英文、数字及下划线组合，长度不超过20个字符
行业分类	策略涉及的行业场景分类
关联审核服务	选择场景需要关联的审核服务

5. 场景创建完成，开始配置场景策略，单击**配置 > 视频审核内容安全**。

6. 在策略详情页面会显示当前系统默认策略，如需更改，单击**编辑**，即可修改当前策略。

参数名称	描述
识别策略配置	根据业务需求，选择是否需要识别不同类型的识别内容。
截帧策略配置	• 审核内容配置：可选全部审核、仅审核视频画面、仅审核音频。 • 图片截帧间隔，设置图片截帧的时间长度，可自行填写设置，默认为1s。 • 音频切片时长，设置音频切片的时间长度，可选15s、30s、60s，也可自行填写设置，默认为30s。 • 审核结果回调配置： ○ 违规回调：仅返回违规审核结果。 ○ 全量回调：返回全部审核结果。
关联自定义库	选择是否需要使用自定义词库用于内容识别，如暂无自定义库，可跳过，或保存当前策略后前往 步骤二：配置自定义词库 。

7. 完成后单击**确认**，完成策略配置。

步骤二：配置自定义词库（可选）

自定义词库用于添加个性化的识别词库。

> **说明**
> 如无需配置自定义词库，可略过此步骤。

1. 登录 内容安全控制台，在左侧导航栏中，选择**名单管理> 关键词名单 > 自定义名单**。
2. 在自定义名单页面，单击**新建词库**，填写文本库名称及按业务需求添加关键词。

参数名称	描述
文本库名称	词库的文字描述，可使用中文、英文、数字及下划线组合，长度不超过32个字符。
处理建议	可选择违规、疑似、放过。 • 违规：确认为不良信息。 • 疑似：可能为不良信息，需要人工识别。 • 放过：业务方认为的正常信息。
匹配模式	可选择精确匹配或模糊匹配。 • 精确匹配：对输入文本进行匹配，匹配对象需完全一致。 • 模糊匹配：可检测变体后的输入词，支持拆分字、形似字、音似字、简繁体、大小写、大写数字等形式的相似词进行匹配。

3. 单击**保存**，即可创建自定义词库。
4. 在自定义词库页面下方的列表中，将显示刚创建的词库，单击**管理**。

> **说明**
> 自定义词库的不同颜色代表不同的屏蔽逻辑，红色代表违规，橙色代表疑似，绿色代表通过。

5. 在词库编辑页面，单击**添加关键词**，选择处理建议，输入关键词，单击**确定**，即可保存关键词至当前词库。

参数名称	描述
识别类型	识别模型对应的违规类型。
关键词	• 每个关键词以换行来确定，单个关键词长度20 个汉字以内。 • 支持以复制粘贴的方式批量导入，单次最多添加关键词2000个，词与词之间用换行符分隔。 • 关键词的添加个数上限为10000个。 • 如需添加组合词，选择关键词类型：组合词，使用&符号对两个或者多个词进行连接，添加格式为“关键词1&关键词2...”，其中关键词只支持纯中文，这些关键词同时出现时则会命中。

说明：
自定义词库配置完成后，可以在 步骤一：配置策略 时将自定义词库与策略关联使用。

步骤三：创建视频内容安全服务

在完成以上步骤后，可以调用**创建视频识别任务（CreateVideoModerationTask）**接口创建直播间视频识别任务，具体方法如下：

- 确保视频符合接口传入的 文件格式要求 。
- 参考任务创建接口的 接口文档说明 ，填入相应的输入参数。
- 若任务创建成功，可通过查询任务接口查询任务相关信息，请参考 创建视频识别任务 进一步了解返回参数示例；若任务创建失败，接口会返回错误码，请参考 业务错误码 和 公共错误码 进行问题排查。

说明
在进行服务接入时，可以使用 API Explorer 工具进行在线调试。

步骤四：获取视频内容安全任务结果

在创建视频识别任务之后，可以调用**查看任务详情（DescribeTaskDetail）**接口查询任务的详细信息，具体方法如下：

- 参考查看任务详情的 接口文档说明 ，填入相应的输入参数。
- 若接口调用成功，您会收到来自接口的响应输出，包含查询的任务的详细信息，可参考 查看任务详情示例 进一步了解返回参数示例。

权限管理
访问控制和权限管理

最近更新时间：2025-02-05 16:07:53

概述

对于腾讯视频内容安全（Video Moderation System，VM）资源，不同企业之间或同企业多团队之间，需要对不同的团队或人员配置不同的访问权限。
您可通过访问管理（Cloud Access Management，CAM）对子账号设置不同的操作权限，使得不同团队或人员能够相互协作。

首先，我们需要先了解几个关键概念：主账号、子账号（用户）和用户组。CAM 的相关术语、配置详细描述请参见访问管理的 词汇表。

主账号

主账号又被称为开发商。用户申请腾讯云账号时，系统会创建一个用于登录腾讯云服务的主账号身份。主账号是腾讯云资源使用计量计费的基本主体。
主账号默认拥有其名下所拥有的资源的完全访问权限，包括访问账单信息，修改用户密码，创建用户和用户组以及访问其他云服务资源等。默认情况下，资源只能被主账号所访问，任何其他用户访问都需要获得主账号的授权。

子账号（用户）和用户组

子账号是由主账号创建的实体，有确定的身份 ID 和身份凭证，拥有登录腾讯云控制台的权限。
子账号默认不拥有资源，必须由所属主账号进行授权。
一个主账号可以创建多个子账号（用户）。
一个子账号可以归属于多个主账号，分别协助多个主账号管理各自的云资源，但同一时刻，一个子账号只能登录到一个主账号下，管理该主账号的云资源。
子账号可以通过控制台切换开发商（主账号），从一个主账号切换到另外一个主账号。
子账号登录控制台时，会自动切换到默认主账号上，并拥有默认主账号所授予的访问权限。
切换开发商之后，子账号会拥有切换到的主账号授权的访问权限，而切换前的主账号授予的访问权限会立即失效。
用户组是多个相同职能的用户（子账号）的集合。您可以根据业务需求创建不同的用户组，为用户组关联适当的策略，以分配不同权限。

操作步骤

授权子账号访问 VM 分为三个步骤：创建子账号、对子账号授予权限、子账号访问 VM 资源。

步骤1：创建子账号

在 CAM 控制台可创建子账号，并配置授予子账号的访问权限。具体操作如下所示：

1. 使用主账号登录 CAM 控制台。
2. 选择**用户 > 用户列表 > 新建用户**，进入新建用户页面。
3. 选择**自定义创建**，选择**可访问资源并接收消息**类型，单击**下一步**。
4. 按照要求填写用户相关信息。
   - **设置用户信息**：输入子用户名称，例如 Sub_user。输入子用户的邮箱，您需要为子用户添加邮箱来获取由腾讯云发出的绑定微信的邮件。
   - **访问方式**：选择编程访问和腾讯云控制台访问。其他配置可按需选择。
5. 填写用户信息完毕后，单击**下一步**，进行身份验证。
6. 身份验证完毕，设置子用户权限。根据系统提供的策略选择，策略配置说明可参考 步骤2 。
7. 设置用户标签，该项为可选项，可按需设置，单击**下一步**。
8. 确认配置信息无误后，单击**完成**即可创建子账号。

步骤2：授予子账号控制台和 API 资源访问权限

根据 步骤1 设置的访问方式：编程访问和腾讯云控制台访问。

编程访问

当使用子账号通过编程（例如 API、SDK 和工具等）访问 VM 资源时需要先获取主账号的 APPID、子账号的 SecretId 和 SecretKey 信息。您可以在访问管理控制台生成子账号的 SecretId 和 SecretKey。

1. 使用主账号登录 CAM 控制台。
2. 选择**用户列表**，进入用户列表页面。
3. 单击**子账号用户名称**，进入子账号信息详情页。

4. 单击 API 密钥页签，并单击**新建密钥**为该子账号创建 SecretId 和 SecretKey。
5. 按照 步骤1 授权 VM 的 CAM 策略 **QcloudVMFullAccess** 后，至此您就可以通过子账号的 SecretId 和 SecretKey 访问 VM 资源。

腾讯云控制台

子用户被授予权限后，可在 子用户登录界面 输入主账号 ID、子用户名和子用户密码登录控制台，并在**云产品**中选择单击**内容安全**，即可进入控制台。
登录控制台必须需授权以下 CAM：

- 图片策略：QcloudIMSReadOnlyAccess
- 文本策略：QcloudTMSReadOnlyAccess
- 音频策略：QcloudIAMSReadOnlyAccess
- 视频策略：QcloudVMReadOnlyAccess（如需使用 服务体验，需授权QcloudVMFullAccess）
- CMS策略：QcloudCMSFullAccess

子账号查询其他子用户数据

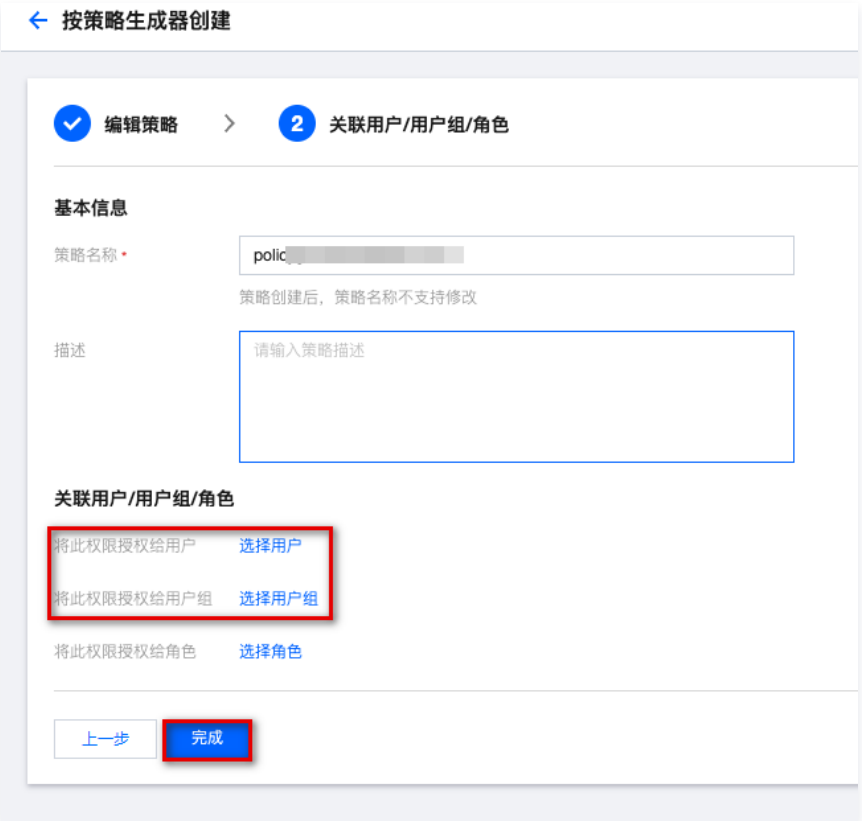
主账号可以查看所有子用户的调用数据，而子账号只能查看自己账号的调用记录。如果想让子账号也能够访问其他用户的信息，可以通过授权 ListUsers 权限实现。当前功能不支持角色 SSO 登录方式。

1. 在 访问管理 > **策略**页面，单击**新建自定义策略**，创建一个自定义策略配置 ListUsers 接口权限并关联对应用户。

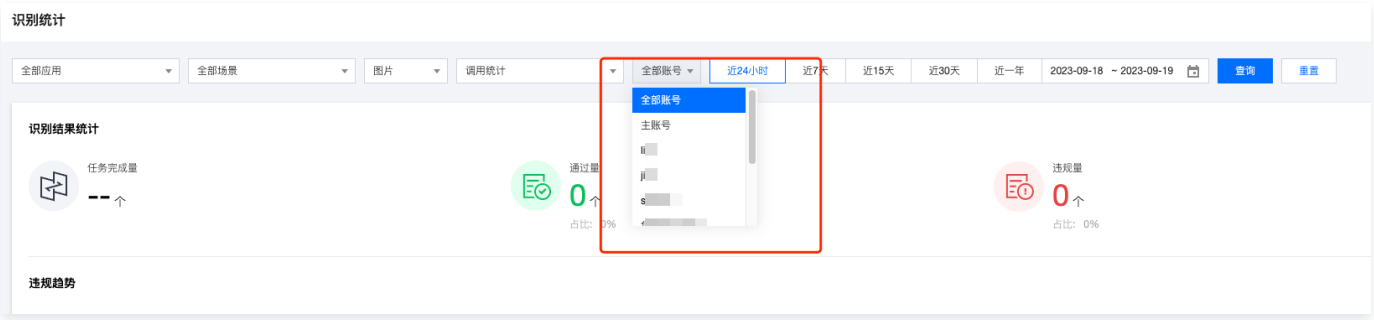
2. 在新的窗口中，选择**按策略生成器创建**。

3. 在编辑策略页面，服务选择**访问管理（cam）**，操作选择 **ListUsers**，资源选择**全部**，单击**下一步**。

4. 在关联用户/用户组/角色页面，输入策略名称和描述，并且授予需要相应权限的用户和用户组，单击**完成**。



5. 授权后，子账号登录控制台后，对应的多个页面的界面查询框部分有用户选择框。并且默认是当前子账户，可以手动选择全部账户，主账户和其他账户，数据权限和主账号权限一致。



热点问题

腾讯云账号的 UIN 和 APPID 是什么？子账号是什么？

1. UIN 即账号 ID，是唯一的且不能修改，APPID 是腾讯云账号的 APPID，是与账号 ID 有唯一对应关系的应用 ID，部分腾讯云产品会使用此 APPID。可在腾讯云控制台的账号信息中心查看 UIN 和 APPID。
2. 子账号是由主账号创建的实体，有确定的身份 ID 和身份凭证，拥有登录腾讯云控制台的权限。子账号默认不拥有资源，必须由所属主账号进行授权。一个主账号可以创建多个子账号。子账号身份 ID 和凭证可以在 腾讯云控制台 查看。

如何查看腾讯云后台子账号？

可以通过**账号登录** 访问管理 > 用户列表 查看已创建的**子账号信息**。

如何获取 API 密钥?

可以通过**主账号登录** 访问管理 > API 密钥管理 查看已创建的**子账号信息** 。

说明:

- AppID 和 SecretID不是一样的。
- 在 云 API 密钥管理页面可以新建 SecretID, 每个账号最多可以创建两个 SecretID，没有有效期。
- 在 API 密钥管理页面获得的密钥，适用腾讯云所有的 API 调用使用。

访问管理 CAM 是什么?

是指帮助客户安全管理对腾讯云产品和资源的访问，默认主账号拥有腾讯云所有权限，客户需要用子账号来调用服务，就需要主账号来给子账号授权对应的CAM权限。否则会返回如下错误:

```
{
    "Response":{
        "Error":{
            "Code":"AuthFailure.UnauthorizedOperation",
            "Message":"You are not authorized to perform this operation. Check your CAM policies, and ensure that you are using the correct access keys. [[request id:ba58220a-0710-4fb4-b7b0-11d4f5a04f94]you are not authorized to perform operation (tms:TextModeration)\nresource (*) has no permission\n]"
        },
        "RequestId":"ba58220a-0710-4fb4-b7b0-11d4f5a04f94"
    }
}
```

子账号授权

最近更新时间：2025-02-05 16:07:53

操作场景

创建用户/用户组时，默认没有任何权限，您可以通过为其关联策略，使用户/用户组获得对应的操作权限。

前提条件

已创建完成 子用户 或 用户组 。

操作步骤

您可以通过策略关联用户/用户组，或者通过用户/用户组关联策略，两种方式操作入口有区别，实现的功能无区别。

通过策略关联用户/用户组

通过策略关联用户

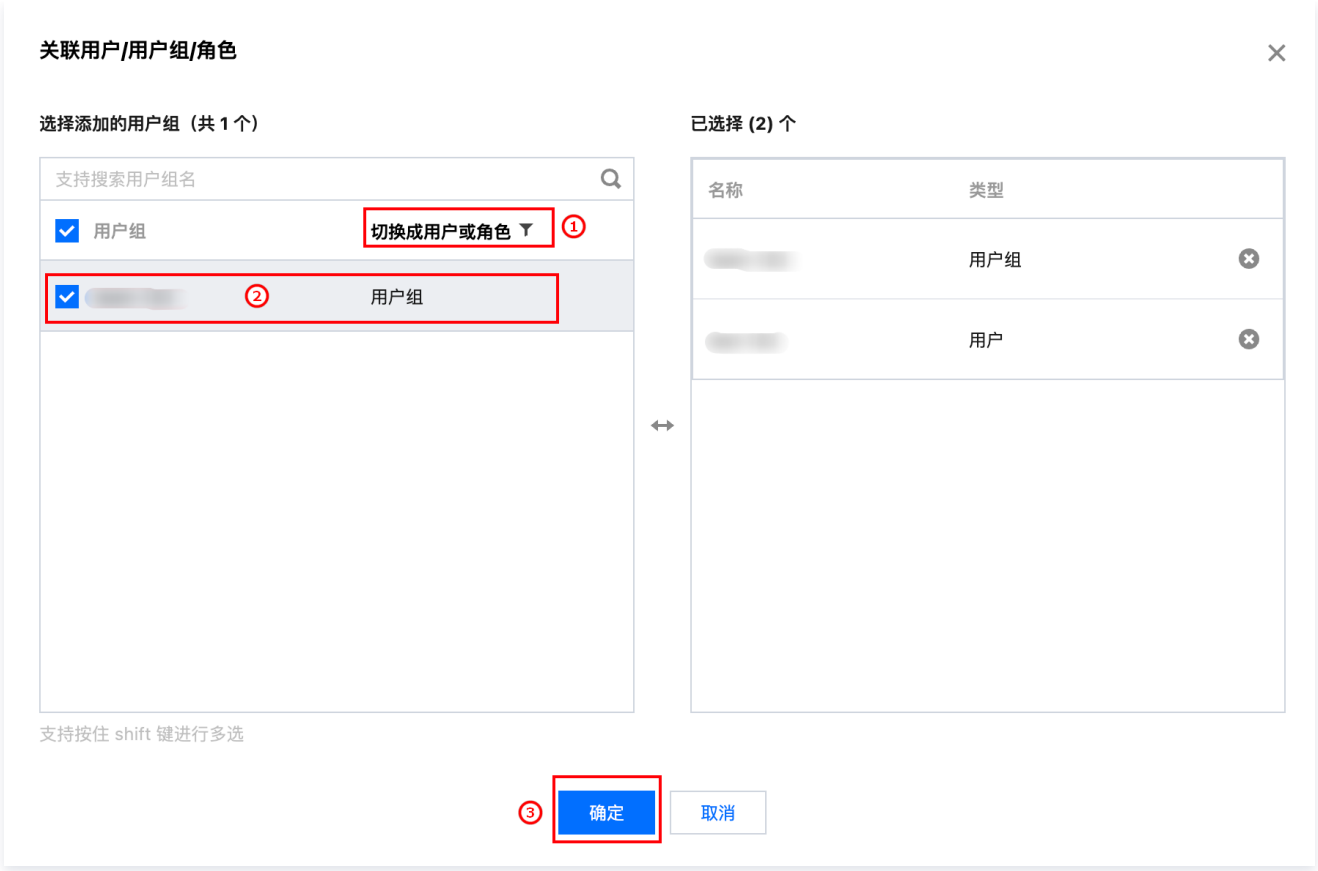
1. 使用腾讯云主账号，登录 访问管理控制台 ，在左侧导航中，单击**策略**。
2. 在策略页面，通过搜索筛选找到**视频内容安全**，选择需要授权的预设策略 QcloudVMFullAccess，单击操作列中的**关联用户/组/角色**。

3. 在“关联用户/用户组/角色”窗口中，勾选要关联的用户，单击**确定**，完成通过策略关联用户操作。

通过策略关联用户组

1. 使用腾讯云主账号，登录 访问管理控制台，在左侧导航中，单击**策略**。
2. 在策略页面，通过搜索筛选找到**视频内容安全**，选择需要授权的预设策略 QcloudVMFullAccess，单击操作列中的**关联用户/组/角色**。

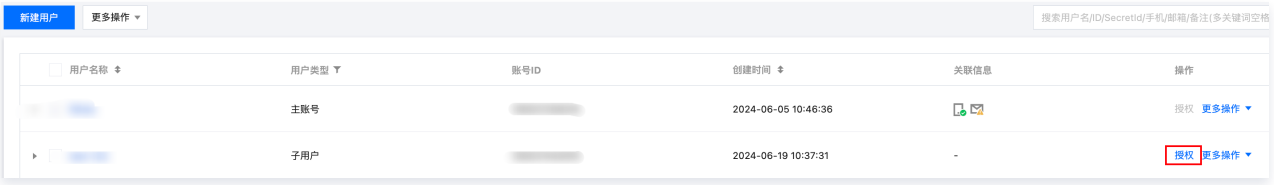
3. 在“关联用户/用户组/角色”窗口中，单击**切换成用户组或角色**，选择**用户组**。
4. 勾选要关联的用户组，单击**确定**，完成通过策略关联用户组操作。



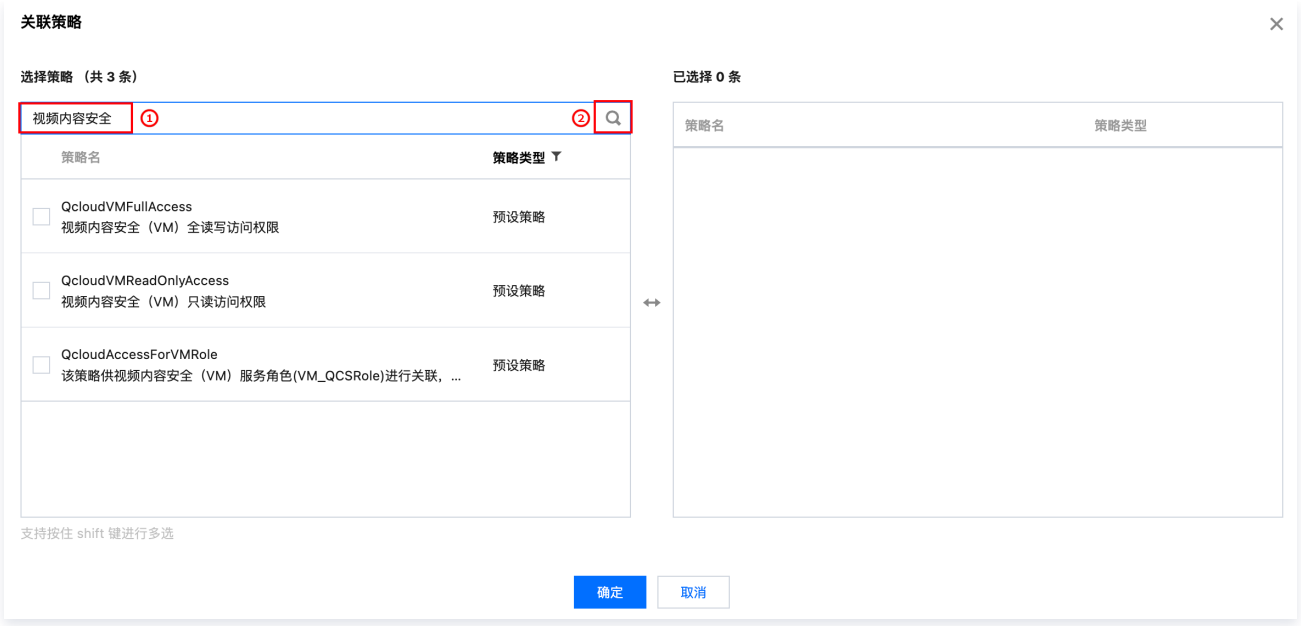
通过用户/用户组关联策略

通过用户关联策略

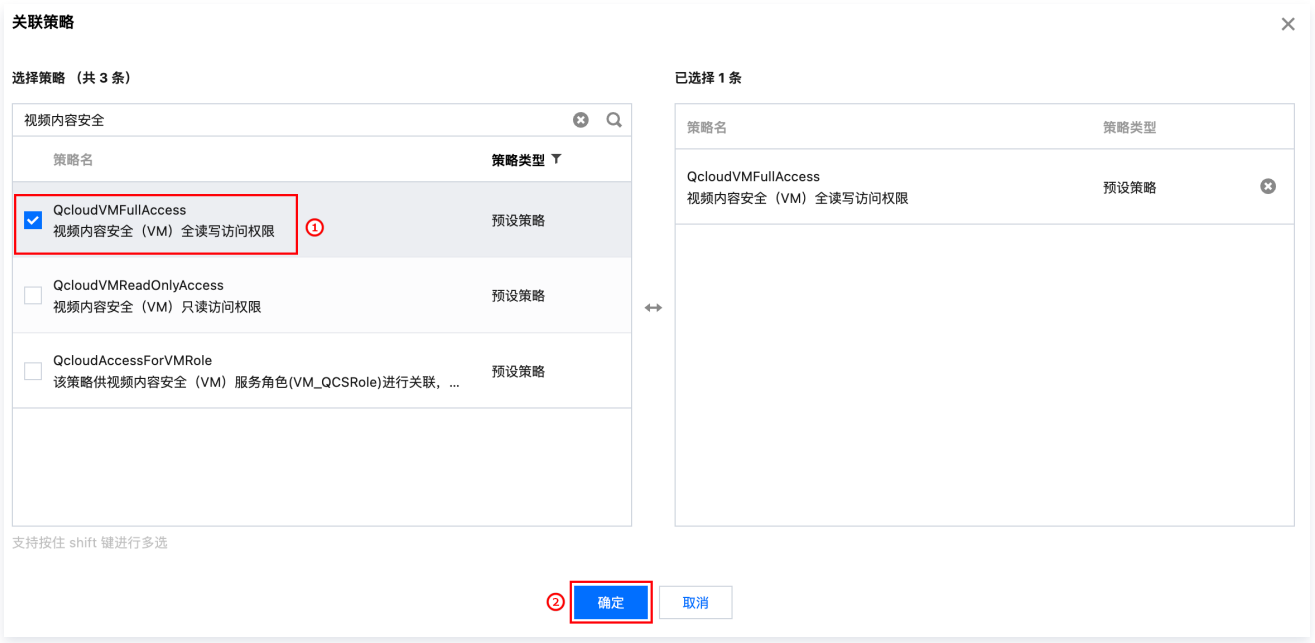
1. 使用腾讯云主账号，登录 访问管理控制台 ，在左侧导航中，单击**用户 > 用户列表**。
2. 在用户列表页面，找到需要授权的用户，单击操作列的**授权**，进入关联策略页面。



3. 在关联策略页面，输入**视频内容安全**，单击🔍。

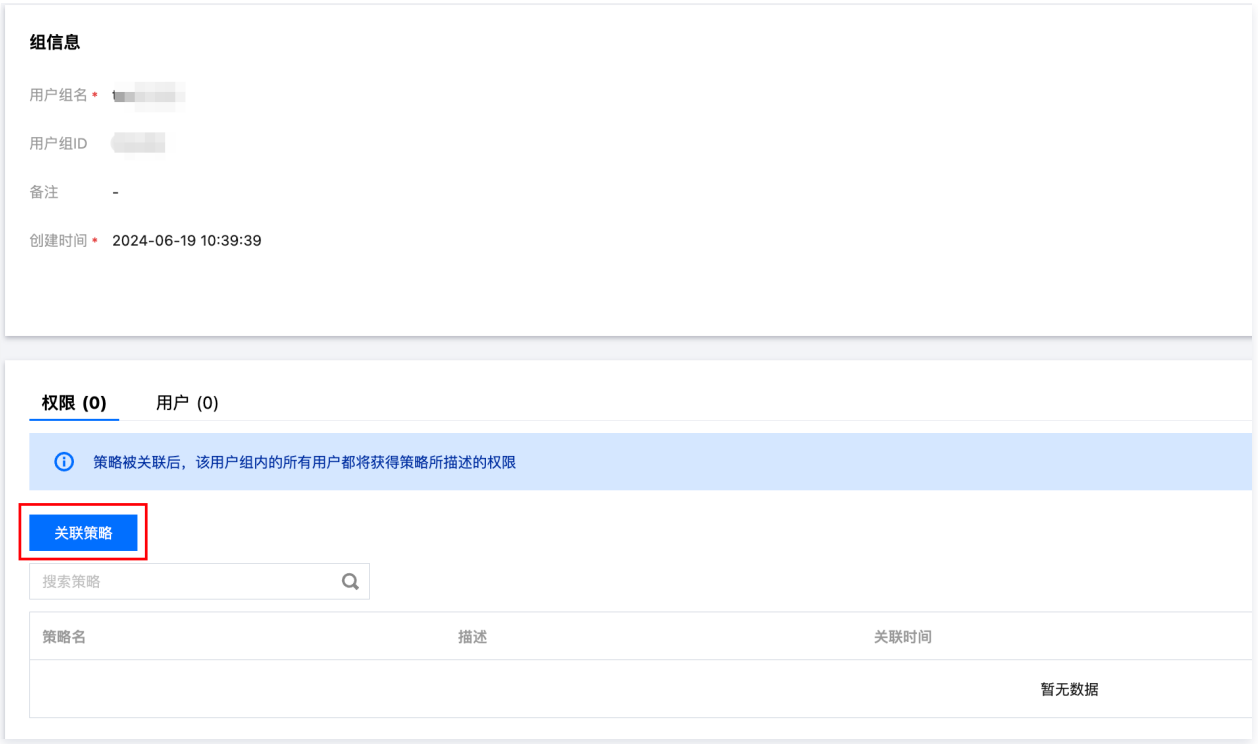


4. 勾选需要授权的策略：**QcloudVMFullAccess**，单击**确定**，完成通过用户关联预设策略操作。

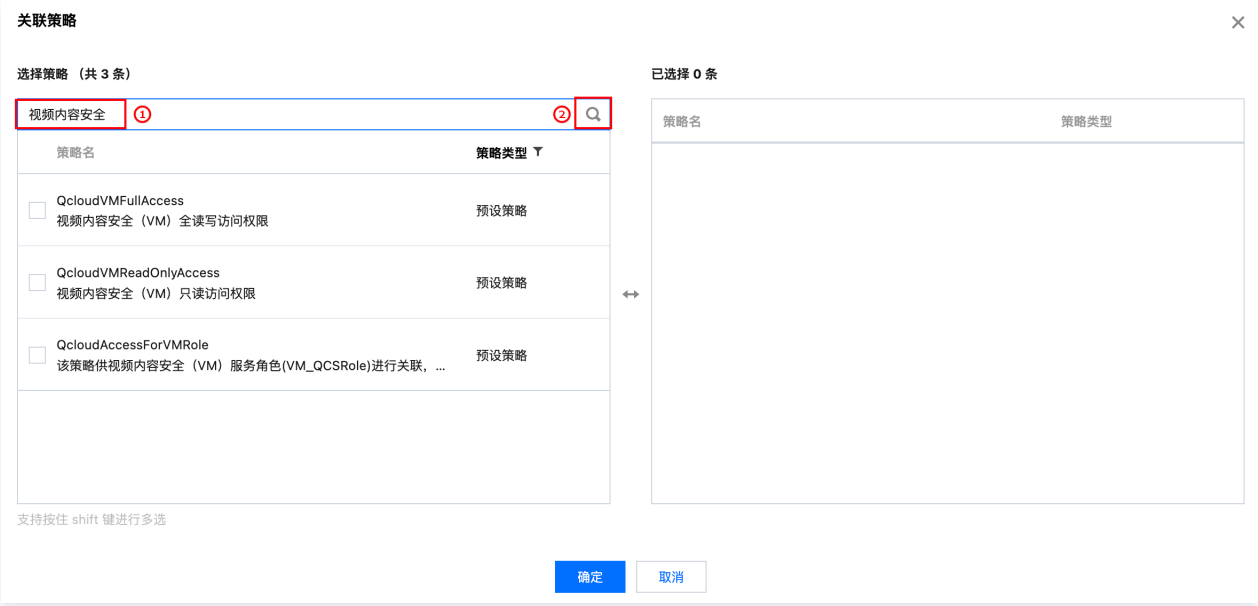


通过用户组关联策略

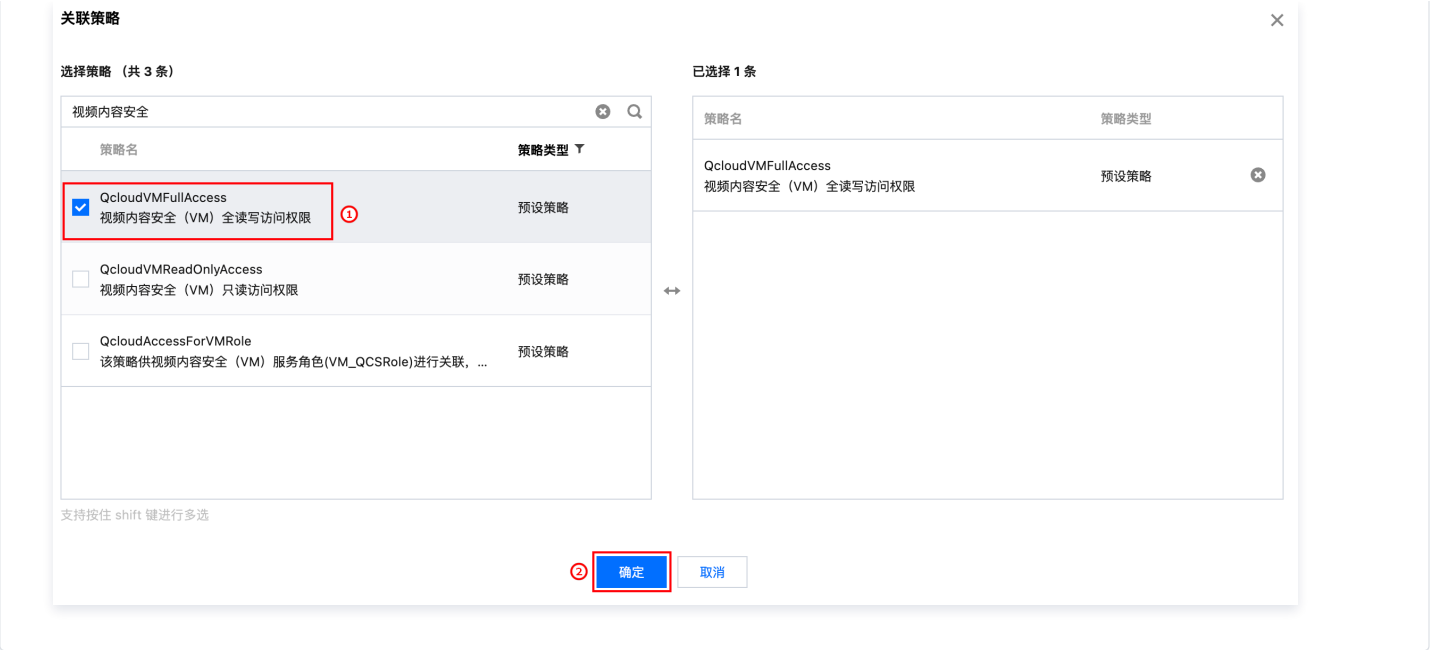
1. 使用腾讯云主账号，登录 访问管理控制台，在左侧导航中，单击**用户组**。
2. 在用户组页面，单击目标**用户组名称**，进入用户组详情页。
3. 在用户组详情页，单击**关联策略**，进入关联策略页面。



4. 在关联策略页面，输入**视频内容安全**，单击  。



5. 勾选需要授权的策略：**QcloudVMFullAccess**，单击**确定**，完成通过用户关联预设策略操作。


关联策略
选择策略 （共 3 条）
已选择 1 条
视频内容安全
策略名
策略类型
QcloudVMFullAccess
视频内容安全（VM）全读写访问权限
预设策略
QcloudVMReadOnlyAccess
视频内容安全（VM）只读访问权限
预设策略
QcloudAccessForVMRole
该策略供视频内容安全（VM）服务角色(VM_QCSRole)进行关联，...
预设策略
支持按住 shift 键进行多选
确定
取消

腾讯云COS资源访问授权

最近更新时间：2025-02-05 16:07:53

1. 建议使用 URL 资源链接提交审核服务。

   如用户需通过 COS 路径送审，**必须手动授权内容审核侧账号**。传参格式如下：

```
{
    "BizType": "default",
    "Tasks": [
        {
            "DataId": "test",
            "Input": {
                "BucketInfo": {
                    "Bucket": "bucket-12xxxxx33",
                    "Region": "ap-guangzhou",
                    "Object": "path/xxx.mp4"
                },
                "Type": "COS"
            }
        }
    ],
    "Type": "VIDEO"
}
```

2. 授权需登录 COS 控制台，在存储桶列表页面，单击目标**存储桶名称**。

3. 在**权限管理 > 存储桶访问权限**页面，单击**添加用户**。
4. 用户类型选择**主账号**，账号 ID 为审核服务侧账号（**100004528167**），权限勾选**数据读取**和**数据写入**，单击**确定**授权访问所需资源的 COS 桶。