视频理解

API 文档

腾讯云

文档目录

API 文档
更新历史

最近更新时间：2024-12-04 01:22:31

第 10 次发布

发布时间：2024-12-04 01:22:24

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- AudioData
  - 新增成员：WebMediaURL
- AudioMetadata
  - 新增成员：BitDepth, ShortFormat
- PersonInfo
  - 新增成员：PersonId
- TextData
  - 新增成员：WebMediaURL
- TextMetadata
  - 新增成员：ShortFormat
- UnknownPerson
  - 新增成员：AuditClass

第 9 次发布

发布时间：2024-07-09 14:27:21

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- CreateVideoSummaryTask
- DescribeUsageAmount
- DescribeVideoSummaryDetail

新增数据结构：

- AsrResult
- ShotInfo
- TTSMode
- TextSegMatchShotScore
- VideoRotationMode

第 8 次发布

发布时间：2022-10-25 19:11:18

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- PersonInfo
  - 新增成员：AppearRect

第 7 次发布

发布时间：2022-09-20 19:36:34

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- DescribeTaskDetail
  - 新增出参：ImageTaskData, AudioTaskData, TextTaskData
- ImportMedia
  - 新增入参：MediaType

新增数据结构：

- AudioData
- AudioMetadata
- ImageData
- ImageLogo
- ImageMetadata
- ImageOcr
- MultiLevelPersonInfo
- Rectf
- TextData
- TextMetadata
- UnknownPerson

修改数据结构：

- MediaFilter
  - 新增成员：MediaType
- MediaInfo
  - 新增成员：MediaType, AudioMetadata, ImageMetadata, TextMetadata
- ShowInfo
  - 新增成员：UnknownPersonSet, MultiLevelPersonInfoSet
- TaskInfo
  - 新增成员：AudioMetadata, ImageMetadata, TextMetadata, Metadata

第 6 次发布

发布时间：2022-04-24 15:20:20

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- MediaInfo
  - 新增成员：CallbackURL
- TaskInfo
  - 新增成员：CallbackURL

第 5 次发布

发布时间：2022-03-07 10:30:32

本次发布包含了以下内容：

改善已有的文档。

新增数据结构：

- ClassifiedPersonInfo
- PersonInfo

修改数据结构：

- ShowInfo
  - 新增成员：ClassifiedPersonInfoSet

第 4 次发布

发布时间：2022-03-02 12:16:47

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DeleteTask
- ModifyCallback
- QueryCallback

修改接口：

- CreateTask
  - 新增入参：Label, CallbackURL
- ImportMedia
  - 新增入参：WriteBackCosPath, Label, CallbackURL

修改数据结构：

- MediaFilter
  - 新增成员：LabelSet
- MediaInfo
  - 新增成员：Label
- TaskFilter
  - 新增成员：LabelSet
- TaskInfo
  - 新增成员：Label

第 3 次发布

发布时间：2022-02-16 08:08:24

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- CustomPersonInfo
  - 新增成员：ImageInfoSet, CreateTime

第 2 次发布

发布时间：2022-01-25 16:20:40

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- AddCustomPersonImage
- CreateCustomCategory
- CreateCustomGroup
- CreateCustomPerson
- CreateDefaultCategories
- DeleteCustomCategory
- DeleteCustomPerson
- DeleteCustomPersonImage
- DescribeCustomCategories
- DescribeCustomGroup
- DescribeCustomPersonDetail
- DescribeCustomPersons
- UpdateCustomCategory
- UpdateCustomPerson

新增数据结构：

- CustomCategory
- CustomPersonFilter
- CustomPersonInfo
- PersonImageInfo

第 1 次发布

发布时间：2021-11-30 14:02:22

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- CreateTask
- DeleteMedia
- DescribeMedia
- DescribeMedias
- DescribeTask
- DescribeTaskDetail
- DescribeTasks
- ImportMedia

新增数据结构：

- AppearIndexPair
- AppearInfo
- AudioInfo
- Data
- L1Tag
- L2Tag
- L3Tag
- MediaFilter

- MediaInfo
- MediaMetadata
- MediaPreknownInfo
- MultiLevelTag
- ShowInfo
- SortBy
- TaskFilter
- TaskInfo
- TextAppearInfo
- TextInfo
- VideoAppearInfo

简介

最近更新时间：2022-10-25 19:11:36

欢迎使用 媒体智能标签 API 3.0 版本。全新的 API 接口文档更加规范和全面，统一的参数风格和公共错误码，统一的 SDK/CLI 版本与 API 文档严格一致，给您带来简单快捷的使用体验。支持全地域就近接入让您更快连接腾讯云产品。更多腾讯云 API 3.0 使用介绍请查看：快速入门

欢迎使用视频智能标签(Intelligent Video Label Detection)。

媒体智能标签(Intelligent Video Label Detection)基于腾讯领先的AI技术和丰富的内容运营经验，提供视频全维度标签提取能力；通过多模态识别算法输出音视图文的内容文本和全维度标签，并针对视频素材进行定制优化，保证标签结果丰富全面，同时具有高准确率、高有效性、高价值等特点，可供媒体、音视频、电商等行业的内容创作，内容管理，内容运营等业务场景快速使用。

API 概览

最近更新时间：2024-08-02 01:47:02

媒资相关接口

接口名称	接口功能	频率限制（次/秒）
ImportMedia	导入媒资文件	20
DescribeMedia	描述媒资文件	20
DescribeMedias	批量描述媒资文件	20
DeleteMedia	删除媒资文件	20

标签任务相关接口

接口名称	接口功能	频率限制（次/秒）
CreateTask	创建任务	20
DescribeTask	描述任务	20
DescribeTasks	批量描述任务	20
DescribeTaskDetail	描述任务与任务结果	20
DeleteTask	删除任务	20

自定义人物库相关接口

接口名称	接口功能	频率限制（次/秒）
CreateCustomGroup	创建自定义人物库	20
DescribeCustomGroup	描述自定义人物库	20
CreateCustomCategory	创建自定义人物分类	20
DescribeCustomCategories	批量描述自定义人物分类	20
UpdateCustomCategory	更新自定义人物分类	20
DeleteCustomCategory	删除自定义分类	20
CreateCustomPerson	创建自定义人物	20
DescribeCustomPersonDetail	描述自定义人物详细信息	20
DescribeCustomPersons	批量描述自定义人物	20
UpdateCustomPerson	更新自定义人物信息	20
DeleteCustomPerson	删除自定义人物	20
AddCustomPersonImage	增加自定义人脸图片	20
DeleteCustomPersonImage	删除自定义人脸	20
CreateDefaultCategories	创建默认自定义人物类型	20

用户管理相关接口

接口名称	接口功能	频率限制（次/秒）
ModifyCallback	编辑回调地址	20
QueryCallback	查询回调设置	20

视频缩编相关接口

接口名称	接口功能	频率限制（次/秒）
CreateVideoSummaryTask	创建视频缩编任务	20
DescribeUsageAmount	视频摘要使用量统计	20
DescribeVideoSummaryDetail	描述视频缩编任务与任务结果	20

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

调用方式
请求结构

最近更新时间：2024-08-28 01:57:47

1. 服务地址

API 支持就近地域接入，本产品就近地域接入域名为 ivld.tencentcloudapi.com ，也支持指定地域域名访问，例如广州地域的域名为 ivld.ap-guangzhou.tencentcloudapi.com 。

推荐使用就近地域接入域名。根据调用接口时客户端所在位置，会自动解析到**最近的**某个具体地域的服务器。例如在广州发起请求，会自动解析到广州的服务器，效果和指定 ivld.ap-guangzhou.tencentcloudapi.com 是一致的。

注意：对时延敏感的业务，建议指定带地域的域名。

注意：域名是 API 的接入点，并不代表产品或者接口实际提供服务的地域。产品支持的地域列表请在调用方式/公共参数文档中查阅，接口支持的地域请在接口文档输入参数中查阅。

目前支持的域名列表为：

接入地域	域名
就近地域接入（推荐，只支持非金融区）	ivld.tencentcloudapi.com
华南地区(广州)	ivld.ap-guangzhou.tencentcloudapi.com
华东地区(上海)	ivld.ap-shanghai.tencentcloudapi.com
华北地区(北京)	ivld.ap-beijing.tencentcloudapi.com
西南地区(成都)	ivld.ap-chengdu.tencentcloudapi.com
西南地区(重庆)	ivld.ap-chongqing.tencentcloudapi.com
港澳台地区(中国香港)	ivld.ap-hongkong.tencentcloudapi.com
亚太东南(新加坡)	ivld.ap-singapore.tencentcloudapi.com
亚太东南(曼谷)	ivld.ap-bangkok.tencentcloudapi.com
亚太南部(孟买)	ivld.ap-mumbai.tencentcloudapi.com
亚太东北(首尔)	ivld.ap-seoul.tencentcloudapi.com
亚太东北(东京)	ivld.ap-tokyo.tencentcloudapi.com
美国东部(弗吉尼亚)	ivld.na-ashburn.tencentcloudapi.com
美国西部(硅谷)	ivld.na-siliconvalley.tencentcloudapi.com
欧洲地区(法兰克福)	ivld.eu-frankfurt.tencentcloudapi.com

2. 通信协议

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

3. 请求方法

支持的 HTTP 请求方法：

- POST（推荐）
- GET

POST 请求支持的 Content-Type 类型：

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

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

4. 字符编码

均使用 `UTF-8` 编码。

公共参数

最近更新时间：2024-11-15 01:44:03

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

公共参数的具体内容会因您使用的签名方法版本不同而有所差异。

使用签名方法 v3 的公共参数

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

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

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

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

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

HTTP GET 请求结构示例：

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

Authorization: TC3-HMAC-SHA256 Credential=AKID*******************************/2018-10-09/cvm/tc3_request, Signed
Headers=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
Content-Type: application/x-www-form-urlencoded
```

```
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1539084154
X-TC-Region: ap-guangzhou
```

HTTP POST（application/json）请求结构示例：

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

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

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

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

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

Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2018-05-30/cvm/tc3_request, Signed
Headers=content-type;host, Signature=582c400e06b5924a6f2b5d7d672d79c15b13162d9279b0855cfba6789a8edb4c
Content-Type: multipart/form-data; boundary=58731222010402
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1527672334
X-TC-Region: ap-guangzhou

--58731222010402
Content-Disposition: form-data; name="Offset"

0
--58731222010402
Content-Disposition: form-data; name="Limit"

10
--58731222010402--
```

使用签名方法 v1 的公共参数

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

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

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

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

HTTP GET 请求结构示例：

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

Host: cvm.tencentcloudapi.com
```

HTTP POST 请求结构示例：

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

Host: cvm.tencentcloudapi.com
Content-Type: application/x-www-form-urlencoded

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

地域列表

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

地域	取值
华东地区（上海）	ap-shanghai

签名方法 v3

最近更新时间：2024-12-25 01:43:20

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

推荐使用 API Explorer　　点击调试

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

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

为什么要进行签名

签名通过以下方式帮助保护请求：

1. 验证请求者的身份
   签名确保请求是由持有有效访问密钥的人发送的。请参阅控制台 云 API 密钥 页面获取密钥相关信息。

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

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

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

申请安全凭证

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

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

申请安全凭证的具体步骤如下：

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

签名版本 v3 签名过程

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

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

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

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

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

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

下面详细解释签名计算过程。

1. 拼接规范请求串

按如下伪代码格式拼接规范请求串（CanonicalRequest）：

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

字段名称	解释
HTTPRequestMethod	HTTP 请求方法（GET、POST ）。此示例取值为 `POST`。
CanonicalURI	URI 参数，API 3.0 固定为正斜杠（/）。
CanonicalQueryString	发起 HTTP 请求 URL 中的查询字符串，对于 POST 请求，固定为空字符串""，对于 GET 请求，则为 URL 中问号（？）后面的字符串内容，例如：Limit=10&Offset=0。 注意：CanonicalQueryString 需要参考 RFC3986 进行 URLEncode 编码（特殊字符编码后需大写字母），字符集 UTF-8。推荐使用编程语言标准库进行编码。
CanonicalHeaders	参与签名的头部信息，至少包含 host 和 content-type 两个头部，也可加入其他头部参与签名以提高自身请求的唯一性和安全性，此示例额外增加了接口名头部。 拼接规则： 1. 头部 key 和 value 统一转成小写，并去掉首尾空格，按照 key:value\n 格式拼接； 2. 多个头部，按照头部 key（小写）的 ASCII 升序进行拼接。 此示例计算结果是 `content-type:application/json; charset=utf-8\nhost:cvm.tencentcloudapi.com\nx-tc-action:describeinstances\n`。 注意：content-type 必须和实际发送的相符合，有些编程语言网络库即使未指定也会自动添加 charset 值，如果签名时和发送时不一致，服务器会返回签名校验失败。
SignedHeaders	参与签名的头部信息，说明此次请求有哪些头部参与了签名，和 CanonicalHeaders 包含的头部内容是一一对应的。content-type 和 host 为必选头部。

字段名称	解释
	拼接规则： 1. 头部 key 统一转成小写； 2. 多个头部 key（小写）按照 ASCII 升序进行拼接，并且以分号（;）分隔。 此示例为 content-type;host;x-tc-action
HashedRequestPayload	请求正文（payload，即 body，此示例为 `{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}`）的哈希值，计算伪代码为 Lowercase(HexEncode(Hash.SHA256(RequestPayload)))，即对 HTTP 请求正文做 SHA256 哈希，然后十六进制编码，最后编码串转换成小写字母。对于 GET 请求，RequestPayload 固定为空字符串。此示例计算结果是 `35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064`。

根据以上规则，示例中得到的规范请求串如下：

```
POST
/

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

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

2. 拼接待签名字符串

按如下格式拼接待签名字符串：

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

字段名称	解释
Algorithm	签名算法，目前固定为 `TC3-HMAC-SHA256`。
RequestTimestamp	请求时间戳，即请求头部的公共参数 X-TC-Timestamp 取值，取当前时间 UNIX 时间戳，精确到秒。此示例取值为 `1551113065`。
CredentialScope	凭证范围，格式为 Date/service/tc3_request，包含日期、所请求的服务和终止字符串（tc3_request）。**Date 为 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致**；service 为产品名，必须与调用的产品域名一致。此示例计算结果是 2019-02-25/cvm/tc3_request。
HashedCanonicalRequest	前述步骤拼接所得规范请求串的哈希值，计算伪代码为 Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))。此示例计算结果是 `7019a55be8395899b900fb5564e4200d984910f34794a27cb3fb7d10ff6a1e84`。

注意：

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

2019-02-25，而不是 2019-02-26。
2. Timestamp 必须是当前系统时间，且需确保系统时间和标准时间是同步的，如果相差超过五分钟则必定失败。如果长时间不和标准时间同步，可能运行一段时间后，请求失败，返回签名过期错误。

根据以上规则，示例中得到的待签名字符串如下：

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

3. 计算签名

1）计算派生签名密钥，伪代码如下：

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

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

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

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

2）计算签名，伪代码如下：

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

此示例计算结果是 `10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f`。

4. 拼接 Authorization

按如下格式拼接 Authorization：

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

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

根据以上规则，示例中得到的值为：

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

最终完整的调用信息如下：

```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2019-02-25/cvm/tc3_request, Signed
Headers=content-type;host;x-tc-action, Signature=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f
Content-Type: application/json; charset=utf-8
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1551113065
X-TC-Region: ap-guangzhou

{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}
```

⚠ 注意：

请求发送时的 HTTP 头部（Header）和请求体（Payload）必须和签名计算过程中的内容完全一致，否则会返回签名不一致错误。可以通过打印实际请求内容，网络抓包等方式对比排查。

签名演示

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

- Python
- Java
- PHP
- Go
- NodeJS
- .NET
- C++
- Ruby

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

- Signature Demo

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

Java

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

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

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

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

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

// ************* 步骤 1：拼接规范请求串 *************
String httpRequestMethod = "POST";
```

```
String canonicalUri = "/";
String canonicalQueryString = "";
String canonicalHeaders = "content-type:application/json; charset=utf-8\n"
+ "host:" + host + "\n" + "x-tc-action:" + action.toLowerCase() + "\n";
String signedHeaders = "content-type;host;x-tc-action";

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

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

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

// ************* 步骤 4: 拼接 Authorization *************
String authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + credentialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
System.out.println(authorization);

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

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

```
}
}
```

Python

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

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

service = "cvm"
host = "cvm.tencentcloudapi.com"
endpoint = "https://" + host
region = "ap-guangzhou"
action = "DescribeInstances"
version = "2017-03-12"
algorithm = "TC3-HMAC-SHA256"
#timestamp = int(time.time())
timestamp = 1551113065
date = datetime.utcfromtimestamp(timestamp).strftime("%Y-%m-%d")
params = {"Limit": 1, "Filters": [{"Values": [u"未命名"], "Name": "instance-name"}]}

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

# ************* 步骤 2：拼接待签名字符串 *************
credential_scope = date + "/" + service + "/" + "tc3_request"
hashed_canonical_request = hashlib.sha256(canonical_request.encode("utf-8")).hexdigest()
string_to_sign = (algorithm + "\n" +
str(timestamp) + "\n" +
credential_scope + "\n" +
hashed_canonical_request)
print(string_to_sign)
```

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

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

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

Golang

```
package main

import (
"crypto/hmac"
"crypto/sha256"
"encoding/hex"
"fmt"
"os"
"strings"
"time"
)

func sha256hex(s string) string {
b := sha256.Sum256([]byte(s))
return hex.EncodeToString(b[:])
}

func hmacsha256(s, key string) string {
hashed := hmac.New(sha256.New, []byte(key))
hashed.Write([]byte(s))
return string(hashed.Sum(nil))
```

```
}

func main() {
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID******************************
secretId := os.Getenv("TENCENTCLOUD_SECRET_ID")
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ******************************
secretKey := os.Getenv("TENCENTCLOUD_SECRET_KEY")
host := "cvm.tencentcloudapi.com"
algorithm := "TC3-HMAC-SHA256"
service := "cvm"
version := "2017-03-12"
action := "DescribeInstances"
region := "ap-guangzhou"
//var timestamp int64 = time.Now().Unix()
var timestamp int64 = 1551113065

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

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

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

// step 4: build authorization
```

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

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

PHP

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

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

```
$canonicalRequest = $httpRequestMethod."\n"
.$canonicalUri."\n"
.$canonicalQueryString."\n"
.$canonicalHeaders."\n"
.$signedHeaders."\n"
.$hashedRequestPayload;
echo $canonicalRequest.PHP_EOL;

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

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

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

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

Ruby

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

# 密钥参数
```

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

service = 'cvm'
host = 'cvm.tencentcloudapi.com'
endpoint = 'https://' + host
region = 'ap-guangzhou'
action = 'DescribeInstances'
version = '2017-03-12'
algorithm = 'TC3-HMAC-SHA256'
# timestamp = Time.now.to_i
timestamp = 1551113065
date = Time.at(timestamp).utc.strftime('%Y-%m-%d')

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

puts canonical_request

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

# ************* 步骤 3：计算签名 *************
digest = OpenSSL::Digest.new('sha256')
secret_date = OpenSSL::HMAC.digest(digest, 'TC3' + secret_key, date)
```

```
secret_service = OpenSSL::HMAC.digest(digest, secret_date, service)
secret_signing = OpenSSL::HMAC.digest(digest, secret_service, 'tc3_request')
signature = OpenSSL::HMAC.hexdigest(digest, secret_signing, string_to_sign)
puts signature

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

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

DotNet

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

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

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

public static Dictionary<String, String> BuildHeaders(string secretid,
```

```
string secretkey, string service, string endpoint, string region,
string action, string version, DateTime date, string requestPayload)
{
string datestr = date.ToString("yyyy-MM-dd");
DateTime startTime = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc);
long requestTimestamp = (long)Math.Round((date - startTime).TotalMilliseconds, MidpointRounding.AwayFromZero) / 1
000;
// ************* 步骤 1：拼接规范请求串 *************
string algorithm = "TC3-HMAC-SHA256";
string httpRequestMethod = "POST";
string canonicalUri = "/";
string canonicalQueryString = "";
string contentType = "application/json";
string canonicalHeaders = "content-type:" + contentType + "; charset=utf-8\n"
+ "host:" + endpoint + "\n"
+ "x-tc-action:" + action.ToLower() + "\n";
string signedHeaders = "content-type;host;x-tc-action";
string hashedRequestPayload = SHA256Hex(requestPayload);
string canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload;
Console.WriteLine(canonicalRequest);

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

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

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

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

```
headers.Add("Host", endpoint);
headers.Add("Content-Type", contentType + "; charset=utf-8");
headers.Add("X-TC-Timestamp", requestTimestamp.ToString());
headers.Add("X-TC-Version", version);
headers.Add("X-TC-Action", action);
headers.Add("X-TC-Region", region);
return headers;
}
public static void Main(string[] args)
{
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
string SECRET_ID = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
string SECRET_KEY = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_KEY");

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

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

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

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

NodeJS

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

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

function getHash(message, encoding = 'hex') {
```

```
const hash = crypto.createHash('sha256')
return hash.update(message).digest(encoding)
}

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

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

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

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

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

const canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload
console.log(canonicalRequest)

// ************* 步骤 2：拼接待签名字符串 *************
const algorithm = "TC3-HMAC-SHA256"
const hashedCanonicalRequest = getHash(canonicalRequest);
```

```
const credentialScope = date + "/" + service + "/" + "tc3_request"
const stringToSign = algorithm + "\n" +
timestamp + "\n" +
credentialScope + "\n" +
hashedCanonicalRequest
console.log(stringToSign)

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

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

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

C++

```
#include <algorithm>
#include <cstdlib>
#include <iostream>
#include <iomanip>
#include <sstream>
#include <string>
#include <stdio.h>
#include <time.h>
#include <openssl/sha.h>
#include <openssl/hmac.h>

using namespace std;

string get_data(int64_t &timestamp)
{
string utcDate;
```

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

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

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

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

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

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

#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX_cleanup(h);
```

```
#else
HMAC_CTX_free(h);
#endif

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

return (ss.str());
}

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

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

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

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

// ************* 步骤 1：拼接规范请求串 *************
string httpRequestMethod = "POST";
string canonicalUri = "/";
string canonicalQueryString = "";
string lower = action;
std::transform(action.begin(), action.end(), lower.begin(), ::tolower);
string canonicalHeaders = string("content-type:application/json; charset=utf-8\n")
```

```
+ "host:" + host + "\n"
+ "x-tc-action:" + lower + "\n";
string signedHeaders = "content-type;host;x-tc-action";
string payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-nam
e\"}]}";
string hashedRequestPayload = sha256Hex(payload);
string canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload;
cout << canonicalRequest << endl;

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

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

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

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

C

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

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

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

void hmac_sha256(const char* key, int key_len,
const char* input, int input_len,
unsigned char* output, unsigned int* output_len)
{
HMAC_CTX *h;
#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX hmac;
HMAC_CTX_init(&hmac);
h = &hmac;
#else
h = HMAC_CTX_new();
#endif

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


#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX_cleanup(h);
```

```
#else
HMAC_CTX_free(h);
#endif

}

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

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

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

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

// ************* 步骤 1：拼接规范请求串 *************
const char* http_request_method = "POST";
const char* canonical_uri = "/";
const char* canonical_query_string = "";
char canonical_headers[100] = {"content-type:application/json; charset=utf-8\nhost:"};
strcat(canonical_headers, host);
```

```
strcat(canonical_headers, "\nx-tc-action:");
char value[100] = {0};
lowercase(action, value);
strcat(canonical_headers, value);
strcat(canonical_headers, "\n");
const char* signed_headers = "content-type;host;x-tc-action";
const char* payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instanc
e-name\"}]}";
char hashed_request_payload[100] = {0};
sha256_hex(payload, hashed_request_payload);

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


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

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

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

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

```
printf("%s\n", authorization);

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

其他语言

- Lua: GitHub
- Swift: GitHub
- Dart: GitHub
- Shell(Bash): GitHub

签名失败

存在以下签名失败的错误码，请根据实际情况处理。

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

签名方法

最近更新时间：2024-12-25 01:43:21

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

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

推荐使用 API Explorer　　点击调试

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

腾讯云 API 会对每个访问请求进行身份验证，即每个请求都需要在公共请求参数中包含签名信息（Signature）以验证请求者身份。
签名信息由安全凭证生成，安全凭证包括 SecretId 和 SecretKey；若用户还没有安全凭证，请前往 云API密钥页面 申请，否则无法调用云 API 接口。

1. 申请安全凭证

在第一次使用云 API 之前，请前往 云 API 密钥页面 申请安全凭证。
安全凭证包括 SecretId 和 SecretKey：

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

申请安全凭证的具体步骤如下：

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

注意：每个账号最多可以拥有两对 SecretId/SecretKey。

2. 生成签名串

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

假设用户的 SecretId 和 SecretKey 分别是：

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

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

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

参数名称	中文	参数值
Action	方法名	DescribeInstances
SecretId	密钥 ID	AKID********************************
Timestamp	当前时间戳	1465185768
Nonce	随机正整数	11886
Region	实例所在区域	ap-guangzhou
InstanceIds.0	待查询的实例 ID	ins-09dx96dg

参数名称	中文	参数值
Offset	偏移量	0
Limit	最大允许输出	20
Version	接口版本号	2017-03-12

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

2.1. 对参数排序

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

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

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

2.2. 拼接请求字符串

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

然后将格式化后的各个参数用"&"拼接在一起，最终生成的请求字符串为:

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

2.3. 拼接签名原文字符串

此步骤生成签名原文字符串。
签名原文字符串由以下几个参数构成:

1. 请求方法: 支持 POST 和 GET 方式，这里使用 GET 请求，注意方法为全大写。
2. 请求主机:查看实例列表(DescribeInstances)的请求域名为：cvm.tencentcloudapi.com。实际的请求域名根据接口所属模块的不同而不同，详见各接口说明。
3. 请求路径: 当前版本云API的请求路径固定为 / 。
4. 请求字符串: 即上一步生成的请求字符串。

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

示例的拼接结果为:

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

2.4. 生成签名串

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

具体代码如下，以 PHP 语言为例：

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

最终得到的签名串为：

```
7RAM2xfNMO9EiVTNmPg06MRnCvQ=
```

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

3. 签名串编码

生成的签名串并不能直接作为请求参数，需要对其进行 URL 编码。

如上一步生成的签名串为 7RAM2xfNMO9EiVTNmPg06MRnCvQ= ，最终得到的签名串请求参数（Signature）为：7RAM2xfNMO9EiVTNmPg06MRnCvQ%3D，它将用于生成最终的请求 URL。

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

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

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

4. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理。

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

5. 签名演示

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

- Python
- Java
- PHP
- Go
- NodeJS
- .NET
- C++
- Ruby

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

- Signature Demo

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

最终输出的 url 可能为：`https://cvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKID********************************&Signature=7RAM2xfNMO9EiVTNmPg06MRnCvQ%3D&Timestamp=1465185768&Version=2017-03-12` 。

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

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

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

Java

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

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

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

public static String getStringToSign(TreeMap<String, Object> params) {
StringBuilder s2s = new StringBuilder("GETcvm.tencentcloudapi.com/?");
// 签名时要求对参数进行字典排序，此处用TreeMap保证顺序
for (String k : params.keySet()) {
```

```
s2s.append(k).append("=").append(params.get(k).toString()).append("&");
}
return s2s.toString().substring(0, s2s.length() - 1);
}

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

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

Python

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

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

import requests

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

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

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

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

Golang

```
package main

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

func main() {
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secretId := os.Getenv("TENCENTCLOUD_SECRET_ID")
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
secretKey := os.Getenv("TENCENTCLOUD_SECRET_KEY")
params := map[string]string{
"Nonce": "11886",
```

```
"Timestamp": strconv.Itoa(1465185768),
"Region": "ap-guangzhou",
"SecretId": secretId,
"Version": "2017-03-12",
"Action": "DescribeInstances",
"InstanceIds.0": "ins-09dx96dg",
"Limit": strconv.Itoa(20),
"Offset": strconv.Itoa(0),
}

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

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

for i := range keys {
k := keys[i]
buf.WriteString(k)
buf.WriteString("=")
buf.WriteString(params[k])
buf.WriteString("&")
}
buf.Truncate(buf.Len() - 1)

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

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

PHP

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

```
$param["Limit"] = 20;
$param["Offset"] = 0;

ksort($param);

$signStr = "GETcvm.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));
echo $signature.PHP_EOL;
// need to install and enable curl extension in php.ini
// $param["Signature"] = $signature;
// $url = "https://cvm.tencentcloudapi.com/?".http_build_query($param);
// echo $url.PHP_EOL;
// $ch = curl_init();
// curl_setopt($ch, CURLOPT_URL, $url);
// $output = curl_exec($ch);
// curl_close($ch);
// echo json_decode($output);
```

Ruby

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

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

method = 'GET'
endpoint = 'cvm.tencentcloudapi.com'
data = {
'Action' => 'DescribeInstances',
'InstanceIds.0' => 'ins-09dx96dg',
'Limit' => 20,
'Nonce' => 11886,
'Offset' => 0,
'Region' => 'ap-guangzhou',
'SecretId' => secret_id,
'Timestamp' => 1465185768, # Time.now.to_i
'Version' => '2017-03-12',
}
sign = method + endpoint + '/?'
params = []
data.sort.each do |item|
```

```
params << "#{item[0]}=#{item[1]}"
end
sign += params.join('&')
digest = OpenSSL::Digest.new('sha1')
data['Signature'] = Base64.encode64(OpenSSL::HMAC.digest(digest, secret_key, sign))
puts data['Signature']

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

DotNet

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

public class Application {
public static string Sign(string signKey, string secret)
{
string signRet = string.Empty;
using (HMACSHA1 mac = new HMACSHA1(Encoding.UTF8.GetBytes(signKey)))
{
byte[] hash = mac.ComputeHash(Encoding.UTF8.GetBytes(secret));
signRet = Convert.ToBase64String(hash);
}
return signRet;
}
public static string MakeSignPlainText(SortedDictionary<string, string> requestParams, string requestMethod, stri
ng requestHost, string requestPath)
{
string retStr = "";
retStr += requestMethod;
retStr += requestHost;
retStr += requestPath;
retStr += "?";
string v = "";
foreach (string key in requestParams.Keys)
{
v += string.Format("{0}={1}&", key, requestParams[key]);
}
retStr += v.TrimEnd('&');
return retStr;
}

public static void Main(string[] args)
{
```

```
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
string SECRET_ID = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
string SECRET_KEY = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_KEY");

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

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

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

NodeJS

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

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

function formatSignString(reqMethod, endpoint, path, strParam){
let strSign = reqMethod + endpoint + path + "?" + strParam.slice(1);
return strSign;
}
function sha1(secretKey, strsign){
let signMethodMap = {'HmacSHA1': "sha1"};
let hmac = crypto.createHmac(signMethodMap['HmacSHA1'], secretKey || "");
return hmac.update(Buffer.from(strsign, 'utf8')).digest('base64')
}
```

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

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

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

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

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

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

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

// 4. 进行url编码并拼接请求url
```

```
// const req_url = get_req_url(params, endpoint)
// console.log(params['Signature'])
// console.log(req_url)
}
main()
```

返回结果

最近更新时间：2024-03-12 01:36:46

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

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

正确返回结果

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

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

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

错误返回结果

若调用失败，其返回值示例如下为：

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

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

公共错误码

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

参数类型

最近更新时间：2022-08-10 06:38:00

目前腾讯云 API 3.0 输入参数和输出参数支持如下几种数据格式：

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

媒资相关接口
导入媒资文件

最近更新时间：2024-12-04 01:22:30

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

将URL指向的媒资视频文件导入系统之中。

请注意，本接口为异步接口。接口返回MediaId仅代表导入视频任务发起，不代表任务完成，您可调用读接口(DescribeMedia/DescribeMedias)接口查询MediaId

URL字段推荐您使用COS地址，其形式为 `https://${Bucket}-${AppId}.cos.${Region}.myqcloud.com/${ObjectKey}` ，其中 `${Bucket}` 为您的COS桶名称，Region为COS桶所在可用区， `${ObjectKey}` 为指向存储在COS桶内的待分析的视频的ObjectKey

另外，目前产品也支持使用外部URL地址，但是当传入URL为非COS地址时，需要您指定额外的WriteBackCosPath以供产品回写结果数据。

分析完成后，本产品将在您的 `${Bucket}` 桶内创建名为 `${ObjectKey}_${task-create-time}` 的目录(`task-create-time` 形式为1970-01-01T08:08:08)并将分析结果将回传回该目录，也即，结构化分析结果(包括图片，JSON等数据)将会写回 `https://${Bucket}-${AppId}.cos.${Region}.myqcloud.com/${ObjectKey}_${task-create-time}` 目录

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　　点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：ImportMedia。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
URL	是	String	待分析视频的URL，目前只支持*不带签名的*COS地址，字段输入内容最大为1KB 示例值：cos://my-release-9327555674/d833e1e4bb2.mp4
MD5	否	String	待分析视频的MD5，为空时不做校验，否则会做MD5校验，长度必须为32 示例值：9207adf18198dd5b165aae0bdd8f709d
Name	否	String	待分析视频的名称，指定后可支持筛选，视频名称的大小长度不能超过64 示例值：新闻30分
WriteBackCosPath	否	String	当非本人外部视频地址导入时，该字段为转存的cos桶地址且不可为空; 示例：https://${Bucket}-${AppId}.cos.${Region}.myqcloud.com/${PathPrefix}/ (注意，cos路径需要以/分隔符结尾)。 推荐采用本主帐号COS桶，如果使用其他帐号COS桶，请确保COS桶可写，否则可导致分析失败 示例值：https://my-release-9327555674.cos.ap-guangzhou.myqcloud.com/d833e1e4bb2.mp4
Label	否	String	自定义标签，可用于查询 示例值：新闻

参数名称	必选	类型	描述
CallbackURL	否	String	媒资导入完成的回调地址，该设置优先级高于控制台全局的设置； 示例值：http://example.com/api/callback
MediaType	否	Integer	媒资文件类型，详细定义参见MediaPreknownInfo.MediaType 默认为2(视频) 示例值：2

3. 输出参数

参数名称	类型	描述
MediaId	String	媒资文件在系统中的ID 示例值："media-2aHsU6sj"
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 导入图片

输入示例

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

{
"URL": "https://ai-media-251202827.cos.ap-guangzhou.myqcloud.com/phye-debug/male.png",
"Name": "male.png",
"MediaType": "1"
}
```

输出示例

```
{
"Response": {
"MediaId": "media-9DfyyCOZ",
"RequestId": "a642b499-9ebd-4601-a57e-a8cf5c2bee49"
}
}
```

示例2 发起导入任务成功

输入示例

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

```
<公共请求参数>

{
"URL": "https://ai-media-1256936300.cos.ap-guangzhou.myqcloud.com/ai-media/test/test-news-6mins.mp4",
"Name": "demo-video-0"
}
```

输出示例

```
{
"Response": {
"MediaId": "media-a1b2c3d4",
"RequestId": "50f3df82-beae-4f5f-9b47-23e8302f62ae"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.InvalidSecretId	SecredId失效。
AuthFailure.MFAFailure	MFA失败。
AuthFailure.SecretIdNotFound	SecretId不存在。
AuthFailure.SignatureExpire	签名已过期。
AuthFailure.SignatureFailure	签名校验失败。

错误码	描述
AuthFailure.TaskFinished	任务已完成。
AuthFailure.TokenFailure	令牌失败。
AuthFailure.UserActivated	用户已激活。
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
FailedOperation.AiTemplateNotExist	匹配的模板不存在。
FailedOperation.CategoryExist	自定义人物分类已存在。
FailedOperation.CategoryLevelChanged	自定义类型层级变化。
FailedOperation.CategoryReferred	自定义人物分类被引用，不能删除。
FailedOperation.CustomGroupAlreadyExist	自定义人物库已存在。
FailedOperation.DBConnectionError	内部DB连接失败。
FailedOperation.DownloadFailed	媒资文件下载失败。
FailedOperation.FeatureAlgoFailed	图片特征提取失败。
FailedOperation.GetCAMTokenFailed	获取CAM临时鉴权失败。
FailedOperation.GetTaskListFailed	获取任务列表失败。
FailedOperation.GetVideoMetadataFailed	获取媒资信息失败。
FailedOperation.ImageNumExceeded	图片数量过多。
FailedOperation.MD5Mismatch	MD5不匹配。
FailedOperation.MediaAlreadyExist	媒资文件已经存在。
FailedOperation.MediaExpired	媒资文件已经过期。
FailedOperation.MediaInUse	媒资正在使用。
FailedOperation.MediaNotReady	媒体文件未就绪。
FailedOperation.MultipleFacesInImage	图片中包含多张人脸。
FailedOperation.NoFaceInImage	图片中不包含人脸。
FailedOperation.OpenChargeFailed	计费开通失败。
FailedOperation.PersonDuplicated	人脸库中存在相似的人脸。
FailedOperation.PersonNotMatched	人脸图片不属于已知人物。
FailedOperation.PersonNumExceeded	自定义人物数量过多。
FailedOperation.QualityAlgoFailed	图片质量分检测失败。
FailedOperation.QualityTooLow	图片质量分过低。
FailedOperation.SnapshotDeserializeFailed	结果快照反序列化失败。
FailedOperation.StopFlowFailed	停止AI工作室任务失败。
FailedOperation.TaskAlreadyExist	存在相同的任务。

错误码	描述
FailedOperation.TaskNotFinished	视频分析未完成。
FailedOperation.TranscodeFailed	转码失败。
FailedOperation.UploadFailed	上传文件失败。
InternalError.DBConnectionError	内部DB连接失败。
InternalError.DBOperationError	内部DB操作错误。
InternalError.InnerError	内部错误。
InternalError.InternalOverflow	自定义人物请求超过限制。
InvalidParameter.InvalidCategoryId	自定义人物类型ID不合法。
InvalidParameter.InvalidFilePath	文件路径不合法。
InvalidParameter.InvalidImage	图片不合法。
InvalidParameter.InvalidImageId	图片ID不合法。
InvalidParameter.InvalidL1Category	一级自定义类型不合法。
InvalidParameter.InvalidL2Category	二级自定义类型不合法。
InvalidParameter.InvalidMD5	MD5不合法。
InvalidParameter.InvalidMediaId	媒体ID不合法。
InvalidParameter.InvalidMediaLabel	MediaLabel无效。
InvalidParameter.InvalidMediaLang	MediaLang无效。
InvalidParameter.InvalidMediaName	媒体名称非法。
InvalidParameter.InvalidMediaPreknownInfo	MediaPreknownInfo无效。
InvalidParameter.InvalidMediaStatus	媒资状态不合法。
InvalidParameter.InvalidMediaType	MediaType无效。
InvalidParameter.InvalidName	名称不合法。
InvalidParameter.InvalidPageNumber	分页序号不合法。
InvalidParameter.InvalidPageSize	分页大小不合法。
InvalidParameter.InvalidParam	输入字段不合法。
InvalidParameter.InvalidPersonId	人物ID不合法。
InvalidParameter.InvalidSortBy	排序字段不合法。
InvalidParameter.InvalidSortOrder	排序方式不合法。
InvalidParameter.InvalidTaskId	任务ID不合法。
InvalidParameter.InvalidTaskName	任务名称不合法。
InvalidParameter.InvalidTaskStatus	任务状态不合法。
InvalidParameter.InvalidURL	URL不合法。
InvalidParameter.InvalidUin	用户Uin无效。
InvalidParameter.NameTooLong	名称超过长度限制。

错误码	描述
InvalidParameter.ParamTooLong	参数超过长度限制。
InvalidParameter.URLNotResolved	输入URL域名无法解析。
InvalidParameter.UnsupportURL	不支持的URL类型。
LimitExceeded.UsageLimitExceeded	使用量超过限制。
RequestLimitExceeded.BatchImportOverflow	批量导入超过限制。
RequestLimitExceeded.ConcurrencyOverflow	同时发起过多任务。
ResourceNotFound.CustomCategoryNotFound	自定义人物类型不存在。
ResourceNotFound.CustomGroupNotFound	自定义人物库不存在。
ResourceNotFound.MediaNotFound	媒资文件不存在。
ResourceNotFound.RecordNotFound	记录不存在。
ResourceNotFound.TaskNotFound	任务不存在。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。
UnsupportedOperation.MediaNotAccessible	媒资文件不可访问。
UnsupportedOperation.TaskNotAccessible	任务不可访问。

描述媒资文件

最近更新时间：2024-12-04 01:22:30

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

描述媒资文件信息，包括媒资状态，分辨率，帧率等。

如果媒资文件未完成导入，本接口将仅输出媒资文件的状态信息；导入完成后，本接口还将输出媒资文件的其他元信息。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeMedia。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
MediaId	是	String	导入媒资返回的媒资ID 示例值："media-2aHsU6sj"

3. 输出参数

参数名称	类型	描述
MediaInfo	MediaInfo	媒资信息 注意：此字段可能返回 null，表示取不到有效值。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 导入完成，返回媒资视频元信息

输入示例

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

{
```

```
"MediaId": "media-ly5qgk6j"
}
```

输出示例

```
{
"Response": {
"MediaInfo": {
"DownLoadURL": "https://ai-media-1256936300.cos.ap-guangzhou.myqcloud.com/ai-media/test/test-beijingninzao-6mins.
mp4",
"FailedReason": "",
"MediaId": "media-ly5qgk6j",
"Metadata": {
"BitRate": 1420,
"Duration": 356.024,
"FPS": 25,
"FileSize": 63195585,
"Height": 720,
"MD5": "b67489b0f5ca7822c94f4b6102ac7d13",
"NumFrames": 8900,
"Width": 1280
},
"Name": "demo-video-0",
"Progress": 0,
"Label": "",
"Status": 8,
"CallbackURL": "http://example.com/api/callback",
"MediaType": 1,
"AudioMetadata": null,
"ImageMetadata": null,
"TextMetadata": null
},
"RequestId": "d4c8bf8e-43ba-482a-a257-b6ac7a6296ec"
}
}
```

示例2 导入中(正在进行转码)，此时返回的视频元信息没有意义

输入示例

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

{
"MediaId": "media-ly5qgk6j"
}
```

输出示例

```
{
"Response": {
"MediaInfo": {
"DownLoadURL": "https://ai-media-1256936300.cos.ap-guangzhou.myqcloud.com/ai-media/test/test-beijingninzao-6mins.
mp4",
"FailedReason": "",
"MediaId": "media-ly5qgk6j",
"Metadata": {
"BitRate": 1420,
"Duration": 356.024,
"FPS": 25,
"FileSize": 63195585,
"Height": 720,
"MD5": "b67489b0f5ca7822c94f4b6102ac7d13",
"NumFrames": 8900,
"Width": 1280
},
"Name": "demo-video-0",
"Progress": 0,
"Label": "",
"Status": 8,
"CallbackURL": "http://example.com/api/callback",
"MediaType": 1,
"AudioMetadata": null,
"ImageMetadata": null,
"TextMetadata": null
},
"RequestId": "d4c8bf8e-43ba-482a-a257-b6ac7a6296ec"
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
FailedOperation.DownloadFailed	媒资文件下载失败。
FailedOperation.MD5Mismatch	MD5不匹配。
FailedOperation.MediaExpired	媒资文件已经过期。
FailedOperation.MediaNotReady	媒体文件未就绪。
FailedOperation.TranscodeFailed	转码失败。
InternalError.InnerError	内部错误。
InvalidParameter.InvalidMediaId	媒体ID不合法。
ResourceNotFound.MediaNotFound	媒资文件不存在。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

批量描述媒资文件

最近更新时间：2024-03-12 01:36:46

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

依照输入条件，描述命中的媒资文件信息，包括媒资状态，分辨率，帧率等。

请注意，本接口最多支持同时描述50个媒资文件

如果媒资文件未完成导入，本接口将仅输出媒资文件的状态信息；导入完成后，本接口还将输出媒资文件的其他元信息。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeMedias。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
PageNumber	是	Integer	分页序号，从1开始 示例值：1
PageSize	是	Integer	每个分页所包含的元素数量，最大为50 示例值：15
MediaFilter	否	MediaFilter	列举过滤条件，相关限制相见MediaFilter
SortBy	否	SortBy	返回结果排序信息，By字段只支持CreateTime

3. 输出参数

参数名称	类型	描述
TotalCount	Integer	满足过滤条件的媒资视频总数量 示例值：100
MediaInfoSet	Array of MediaInfo	满足过滤条件的媒资信息 注意：此字段可能返回 null，表示取不到有效值。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 列举前2个成功完成的媒资文件

输入示例

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

{
"MediaFilter": {
"StatusSet": [
8
]
},
"PageNumber": 1,
"PageSize": 2
}
```

输出示例

```
{
"Response": {
"MediaInfoSet": [
{
"DownLoadURL": "https://ai-media-1256936300.cos.ap-guangzhou.myqcloud.com/ai-media/test-news.mov",
"FailedReason": "",
"MediaId": "media-es88zd2m",
"Metadata": {
"BitRate": 258778,
"Duration": 43.211,
"FPS": 30,
"FileSize": 1397761,
"Height": 480,
"MD5": "6348474d783a56de32b4b2ac42e74d79",
"NumFrames": 0,
"Width": 848
},
"Name": "demo-video-1",
"Progress": 0,
"Status": 8
},
{
"DownLoadURL": "https://ai-media-1256936300.cos.ap-guangzhou.myqcloud.com/ai-media/test-news-1.mov",
"FailedReason": "",
"MediaId": "media-t9Igr9jf",
"Metadata": {
"BitRate": 258778,
"Duration": 43.211,
"FPS": 30,
"FileSize": 1397761,
"Height": 480,
"MD5": "6348474d783a56de32b4b2ac42e74d79",
```

```
"NumFrames": 0,
"Width": 848
},
"Name": "demo-video-3.mov",
"Progress": 0,
"Status": 8
}
],
"RequestId": "d56d17ca-fa6f-4040-8627-328a08e7a857",
"TotalCount": 11
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
FailedOperation.MediaAlreadyExist	媒资文件已经存在。
InternalError.DBOperationError	内部DB操作错误。
InternalError.InnerError	内部错误。
InvalidParameter.InvalidMD5	MD5不合法。

错误码	描述
InvalidParameter.InvalidMediaId	媒体ID不合法。
InvalidParameter.InvalidMediaName	媒体名称非法。
InvalidParameter.InvalidMediaStatus	媒资状态不合法。
InvalidParameter.InvalidMediaType	MediaType无效。
InvalidParameter.InvalidName	名称不合法。
InvalidParameter.InvalidPageNumber	分页序号不合法。
InvalidParameter.InvalidPageSize	分页大小不合法。
InvalidParameter.InvalidSortBy	排序字段不合法。
InvalidParameter.InvalidSortOrder	排序方式不合法。
InvalidParameter.InvalidURL	URL不合法。
InvalidParameter.InvalidUin	用户Uin无效。
InvalidParameter.NameTooLong	名称超过长度限制。
InvalidParameter.UnsupportURL	不支持的URL类型。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

删除媒资文件

最近更新时间：2024-12-04 01:22:30

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

将MediaId对应的媒资文件从系统中删除。

请注意，本接口仅删除媒资文件，媒资文件对应的视频分析结果不会被删除。如您需要删除结构化分析结果，请调用DeleteTask接口。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DeleteMedia。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
MediaId	是	String	媒资文件在系统中的ID 示例值："media-2aHsU6sj"

3. 输出参数

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

4. 示例

示例1 成功删除媒资文件

输入示例

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

{
"MediaId": "media-2aHsU6sj"
}
```

输出示例

```
{
"Response": {
"RequestId": "4d608fa8-66a3-4a25-94ce-1b904d4807f3"
}
}
```

示例2 删除媒资文件失败-媒资文件不存在

输入示例

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

{
"MediaId": "media-2aHsU6sj"
}
```

输出示例

```
{
"Response": {
"RequestId": "4d608fa8-66a3-4a25-94ce-1b904d4807f3",
"Error": {
"Code": "ResourceNotFound.MediaNotFound",
"Message": "媒资不存在"
}
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee

- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
FailedOperation.MediaAlreadyExist	媒资文件已经存在。
FailedOperation.MediaInUse	媒资正在使用。
InvalidParameter	参数错误。
InvalidParameter.InvalidMediaId	媒体ID不合法。
InvalidParameter.InvalidMediaStatus	媒资状态不合法。
ResourceNotFound.MediaNotFound	媒资文件不存在。
ResourceNotFound.RecordNotFound	记录不存在。
UnauthorizedOperation	未授权操作。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。
UnsupportedOperation.MediaNotAccessible	媒资文件不可访问。

标签任务相关接口
创建任务

最近更新时间：2024-12-04 01:22:29

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

创建智能标签任务。

请注意，本接口为异步接口，**返回TaskId只代表任务创建成功，不代表任务执行成功**。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：CreateTask。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
MediaId	是	String	媒资文件ID 示例值："media-2aHsU6sj"
MediaPreknownInfo	是	MediaPreknownInfo	媒资素材先验知识，相关限制参考MediaPreknownInfo
TaskName	否	String	任务名称，最长100个中文字符 示例值："新闻30分"
UploadVideo	否	Boolean	是否上传转码后的视频，仅设置true时上传，默认为false 示例值：false
Label	否	String	自定义标签，可用于查询 示例值：新闻
CallbackURL	否	String	任务分析完成的回调地址，该设置优先级高于控制台全局的设置； 示例值：http://example.com/api/callback

3. 输出参数

参数名称	类型	描述
TaskId	String	智能标签视频分析任务ID 示例值："task-g8lqk737"
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 输入普通话新闻素材，发起智能标签任务

输入示例

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

{
"MediaPreknownInfo": {
"MediaLang": 1,
"MediaSecondLabel": 0,
"MediaLabel": 1,
"MediaType": 2
},
"TaskName": "demo-task-0",
"MediaId": "media-2aHsU6sj",
"UploadVideo": false
}
```

输出示例

```
{
"Response": {
"RequestId": "2713de41-b294-429b-ac13-2c3f5c49b1db",
"TaskId": "task-g8lqk737"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
FailedOperation.AiTemplateNotExist	匹配的模板不存在。
FailedOperation.MediaExpired	媒资文件已经过期。
FailedOperation.MediaNotReady	媒体文件未就绪。
FailedOperation.TaskAlreadyExist	存在相同的任务。
FailedOperation.TaskNotFinished	视频分析未完成。
InternalError.DBOperationError	内部DB操作错误。
InternalError.InnerError	内部错误。
InvalidParameter.InvalidMediaId	媒体ID不合法。
InvalidParameter.InvalidMediaLabel	MediaLabel无效。
InvalidParameter.InvalidMediaLang	MediaLang无效。
InvalidParameter.InvalidMediaName	媒体名称非法。
InvalidParameter.InvalidMediaType	MediaType无效。
InvalidParameter.InvalidName	名称不合法。
InvalidParameter.InvalidParam	输入字段不合法。
InvalidParameter.InvalidURL	URL不合法。
InvalidParameter.NameTooLong	名称超过长度限制。
InvalidParameter.ParamTooLong	参数超过长度限制。
InvalidParameter.URLNotResolved	输入URL域名无法解析。
ResourceNotFound.MediaNotFound	媒资文件不存在。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

描述任务

最近更新时间：2024-12-04 01:22:29

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

描述智能标签任务进度。

请注意，**此接口仅返回任务执行状态信息，不返回任务执行结果**

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeTask。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
TaskId	是	String	CreateTask返回的任务ID 示例值："task-g8lqk737"

3. 输出参数

参数名称	类型	描述
TaskInfo	TaskInfo	任务信息，详情参见TaskInfo的定义 注意：此字段可能返回 null，表示取不到有效值。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 任务执行中

输入示例

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

{
```

```
"TaskId": "task-g8lqk734"
}
```

输出示例


```
{
"Response": {
"RequestId": "a3010447-5003-4358-9b85-7c81a98b1c8d",
"TaskInfo": {
"MediaName": "本地新闻",
"Label": "",
"CallbackURL": "http://example.com/api/callback",
"FailedReason": "",
"MediaId": "media-2aHsU6sj",
"MediaPreknownInfo": {
"MediaLabel": 1,
"MediaLang": 1,
"MediaSecondLabel": 0,
"MediaType": 2
},
"TaskCreateTime": "2006-01-02T15:04:05Z07:00",
"TaskId": "task-g8lqk734",
"TaskName": "demo-task-0",
"TaskProgress": 2,
"TaskStartTime": "2006-01-02T15:04:05Z07:00",
"TaskStatus": 2,
"TaskTimeCost": 0,
"AudioMetadata": null,
"ImageMetadata": null,
"TextMetadata": null,
"Metadata": null
}
}
}
```


示例2 任务执行完成

输入示例

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

{
"TaskId": "task-g8lqk734"
}
```

输出示例

```
{
"Response": {
"RequestId": "af5fa625-286e-431d-b52c-6c577048e987",
"TaskInfo": {
"MediaName": "本地新闻",
"Label": "",
"CallbackURL": "http://example.com/api/callback",
"FailedReason": "",
"MediaId": "media-2aHsU6sj",
"MediaPreknownInfo": {
"MediaLabel": 1,
"MediaLang": 1,
"MediaSecondLabel": 0,
"MediaType": 2
},
"TaskCreateTime": "2006-01-02T15:04:05Z07:00",
"TaskId": "task-g8lqk734",
"TaskName": "demo-task-0",
"TaskProgress": 0,
"TaskStartTime": "2006-01-02T15:04:05Z07:00",
"TaskStatus": 8,
"TaskTimeCost": 114,
"AudioMetadata": null,
"ImageMetadata": null,
"TextMetadata": null,
"Metadata": null
}
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
InternalError.InnerError	内部错误。
InvalidParameter.InvalidTaskId	任务ID不合法。
ResourceNotFound.RecordNotFound	记录不存在。
ResourceNotFound.TaskNotFound	任务不存在。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

批量描述任务

最近更新时间：2024-11-29 01:23:48

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

依照输入条件，描述命中的任务信息，包括任务创建时间，处理时间信息等。

请注意，本接口最多支持同时描述50个任务信息

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeTasks。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
PageNumber	否	Integer	分页序号，从1开始 示例值：1
PageSize	否	Integer	每个分页所包含的元素数量，最大为50 示例值：15
TaskFilter	否	TaskFilter	任务过滤条件，相关限制参见TaskFilter
SortBy	否	SortBy	返回结果排序信息，By字段只支持CreateTimeStamp

3. 输出参数

参数名称	类型	描述
TotalCount	Integer	满足过滤条件的任务总数量 示例值：1
TaskInfoSet	Array of TaskInfo	满足过滤条件的任务数组 注意：此字段可能返回 null，表示取不到有效值。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 描述前5个完成的任务

输入示例

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

{
"PageNumber": 1,
"PageSize": 5,
"TaskFilter": {
"TaskStatusSet": [
8
]
}
}
```

输出示例


```
{
"Response": {
"RequestId": "dcbd578a-caea-487a-b761-d49c9b7f2ac3",
"TaskInfoSet": [
{
"FailedReason": "",
"MediaId": "media-fdk9hp2v",
"MediaPreknownInfo": {
"MediaLabel": 1,
"MediaLang": 1,
"MediaSecondLabel": 1,
"MediaType": 2
},
"TaskCreateTime": 1634014749,
"TaskId": "task-ctl3bk8k",
"TaskName": "TaskNameTest",
"TaskProgress": 0,
"TaskStartTime": 1634014751,
"TaskStatus": 8,
"TaskTimeCost": 111
},
{
"FailedReason": "",
"MediaId": "media-t9Igr9jf",
"MediaPreknownInfo": {
"MediaLabel": 1,
"MediaLang": 1,
"MediaSecondLabel": 0,
"MediaType": 2
},
"TaskCreateTime": 1634017967,
"TaskId": "task-wodahg5t",
"TaskName": "test-task-0",
```

```
"TaskProgress": 0,
"TaskStartTime": 1634017971,
"TaskStatus": 8,
"TaskTimeCost": 131
},
{
"FailedReason": "",
"MediaId": "media-fdk9hp2u",
"MediaPreknownInfo": {
"MediaLabel": 1,
"MediaLang": 1,
"MediaSecondLabel": 1,
"MediaType": 2
},
"TaskCreateTime": 1634019460,
"TaskId": "task-qq38azcb",
"TaskName": "TaskNameTest",
"TaskProgress": 0,
"TaskStartTime": 1634019464,
"TaskStatus": 8,
"TaskTimeCost": 138
},
{
"FailedReason": "",
"MediaId": "media-fdk9hp2u",
"MediaPreknownInfo": {
"MediaLabel": 1,
"MediaLang": 1,
"MediaSecondLabel": 1,
"MediaType": 2
},
"TaskCreateTime": 1634019990,
"TaskId": "task-kzjesg12",
"TaskName": "TaskNameTest",
"TaskProgress": 0,
"TaskStartTime": 1634019994,
"TaskStatus": 8,
"TaskTimeCost": 148
},
{
"FailedReason": "",
"MediaId": "media-ly5qgk6j",
"MediaPreknownInfo": {
"MediaLabel": 1,
"MediaLang": 1,
"MediaSecondLabel": 1,
"MediaType": 2
},
"TaskCreateTime": 1634021166,
"TaskId": "task-krmtkhia",
"TaskName": "demo-task-0",
"TaskProgress": 0,
"TaskStartTime": 1634021170,
```

```
"TaskStatus": 8,
"TaskTimeCost": 111
}
],
"TotalCount": 17
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
FailedOperation.GetTaskListFailed	获取任务列表失败。
InternalError.InnerError	内部错误。
InvalidParameter.InvalidPageNumber	分页序号不合法。
InvalidParameter.InvalidPageSize	分页大小不合法。
InvalidParameter.InvalidParam	输入字段不合法。
InvalidParameter.InvalidSortBy	排序字段不合法。
InvalidParameter.InvalidTaskId	任务ID不合法。
InvalidParameter.InvalidTaskName	任务名称不合法。

错误码	描述
InvalidParameter.InvalidTaskStatus	任务状态不合法。
InvalidParameter.InvalidUin	用户Uin无效。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

描述任务与任务结果

最近更新时间：2024-12-10 01:41:33

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

描述任务信息，如果任务成功完成，还将返回任务结果

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeTaskDetail。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
TaskId	是	String	创建任务返回的TaskId 示例值："task-g8lqk737"

3. 输出参数

参数名称	类型	描述
TaskInfo	TaskInfo	任务信息，不包含任务结果 注意：此字段可能返回 null，表示取不到有效值。
TaskData	Data	视频任务结果数据，只在视频任务结束时返回 注意：此字段可能返回 null，表示取不到有效值。
ImageTaskData	ImageData	图片任务结果数据，只在图片任务结束时返回 注意：此字段可能返回 null，表示取不到有效值。
AudioTaskData	AudioData	音频任务结果数据，只在音频任务结束时返回 注意：此字段可能返回 null，表示取不到有效值。
TextTaskData	TextData	文本任务结果数据，只在文本任务结束时返回 注意：此字段可能返回 null，表示取不到有效值。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 描述任务与任务结果

输入示例

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

{
"TaskId": "task-l0ae7t96"
}
```

输出示例


```
{
"Response": {
"AudioTaskData": null,
"ImageTaskData": null,
"TextTaskData": null,
"RequestId": "a18b2219-445f-43b9-b844-077d72ebcb85",
"TaskData": {
"ShowInfo": {
"AudioInfoSet": [
{
"Content": "这是我一四年的一件作品，制作的运行是济南的一处职工宿舍楼，材料主要是泡沫板，木板完了指导制作，周期两个月时间。",
"EndTimeStamp": 51.882,
"StartTimeStamp": 0.36,
"Tag": ""
}
],
"ClassifiedPersonInfoSet": [],
"Column": "",
"CoverImageURL": "https://ai-media-251202827.cos.ap-guangzhou.myqcloud.com/%E6%98%9F%E8%A7%86%E9%A2%91.mp4_2021-11-04T12:14:17/cover/show_g8rktHiE.jpg",
"Date": "",
"FrameTagSet": {
"AppearInfo": {
"AudioAppearSet": [],
"TextAppearSet": [],
"VideoAppearSet": [
{
"EndTimeStamp": 34.84,
"ImageURL": "https://ai-media-251202827.cos.ap-guangzhou.myqcloud.com/%E6%98%9F%E8%A7%86%E9%A2%91.mp4_2021-11-04T12:14:17/cover/lens_popjK1fY.jpg",
"StartTimeStamp": 32.56
},
{
"EndTimeStamp": 39.52,
"ImageURL": "https://ai-media-251202827.cos.ap-guangzhou.myqcloud.com/%E6%98%9F%E8%A7%86%E9%A2%91.mp4_2021-11-04T12:14:17/cover/lens_7Mu2jPLZ.jpg",
"StartTimeStamp": 34.88
},
{
```

```
"EndTimeStamp": 41.64,
"ImageURL": "https://ai-media-251202827.cos.ap-guangzhou.myqcloud.com/%E6%98%9F%E8%A7%86%E9%A2%91.mp4_2021-11-04T12:14:17/cover/lens_KHT2pnwl.jpg",
"StartTimeStamp": 39.56
},
{
"EndTimeStamp": 45.44,
"ImageURL": "https://ai-media-251202827.cos.ap-guangzhou.myqcloud.com/%E6%98%9F%E8%A7%86%E9%A2%91.mp4_2021-11-04T12:14:17/cover/lens_GGvu3x4d.jpg",
"StartTimeStamp": 41.68
},
{
"EndTimeStamp": 49.56,
"ImageURL": "https://ai-media-251202827.cos.ap-guangzhou.myqcloud.com/%E6%98%9F%E8%A7%86%E9%A2%91.mp4_2021-11-04T12:14:17/cover/lens_QAt3s6nW.jpg",
"StartTimeStamp": 45.48
},
{
"EndTimeStamp": 39.52,
"ImageURL": "https://ai-media-251202827.cos.ap-guangzhou.myqcloud.com/%E6%98%9F%E8%A7%86%E9%A2%91.mp4_2021-11-04T12:14:17/cover/lens_7Mu2jPLZ.jpg",
"StartTimeStamp": 34.88
},
{
"EndTimeStamp": 39.52,
"ImageURL": "https://ai-media-251202827.cos.ap-guangzhou.myqcloud.com/%E6%98%9F%E8%A7%86%E9%A2%91.mp4_2021-11-04T12:14:17/cover/lens_7Mu2jPLZ.jpg",
"StartTimeStamp": 34.88
}
]
},
"TagSet": [
{
"AppearIndexPairSet": [],
"FirstAppear": 0,
"L2TagSet": [
{
"AppearIndexPairSet": [],
"FirstAppear": 0,
"L3TagSet": [
{
"AppearIndexPairSet": [
{
"AppearIndex": 3,
"Index": 0
},
{
"AppearIndex": 3,
"Index": 1
},
{
"AppearIndex": 3,
"Index": 2
```

```
},
{
"AppearIndex": 3,
"Index": 3
},
{
"AppearIndex": 3,
"Index": 4
}
],
"FirstAppear": 3,
"Name": "建筑物"
}
],
"Name": "城市景观"
},
{
"AppearIndexPairSet": [],
"FirstAppear": 0,
"L3TagSet": [
{
"AppearIndexPairSet": [
{
"AppearIndex": 3,
"Index": 5
}
],
"FirstAppear": 3,
"Name": "农村风貌"
}
],
"Name": "农业场景"
}
],
"Name": "场景"
},
{
"AppearIndexPairSet": [],
"FirstAppear": 0,
"L2TagSet": [
{
"AppearIndexPairSet": [],
"FirstAppear": 0,
"L3TagSet": [
{
"AppearIndexPairSet": [
{
"AppearIndex": 3,
"Index": 6
}
],
"FirstAppear": 3,
"Name": "自行车"
```

```
}
],
"Name": "陆地交通工具"
}
],
"Name": "交通工具"
}
]
},
"Logo": "",
"MediaClassifierSet": [
"知识科普"
],
"Source": "",
"SummarySet": [],
"SummaryTagSet": [],
"TextInfoSet": [
{
"Content": "这是我2014年的一个作品",
"EndTimeStamp": 39.52,
"StartTimeStamp": 34.88,
"Tag": "字幕"
},
{
"Content": "制作的原型是济南的一处宿舍楼",
"EndTimeStamp": 39.52,
"StartTimeStamp": 34.88,
"Tag": "字幕"
},
{
"Content": "材料主要是泡沫板木板瓦楞制等",
"EndTimeStamp": 45.44,
"StartTimeStamp": 41.68,
"Tag": "字幕"
},
{
"Content": "制作周期2个月时间",
"EndTimeStamp": 49.56,
"StartTimeStamp": 45.48,
"Tag": "字幕"
}
],
"TextTagSet": {
"AppearInfo": {
"AudioAppearSet": [
{
"EndPosition": 37,
"Index": 0,
"StartPosition": 34
},
{
"EndPosition": 28,
"Index": 0,
```

```
"StartPosition": 25
},
{
"EndPosition": 25,
"Index": 0,
"StartPosition": 23
},
{
"EndPosition": 20,
"Index": 0,
"StartPosition": 18
}
],
"TextAppearSet": [],
"VideoAppearSet": []
},
"TagSet": [
{
"AppearIndexPairSet": [],
"FirstAppear": 0,
"L2TagSet": [
{
"AppearIndexPairSet": [],
"FirstAppear": 0,
"L3TagSet": [
{
"AppearIndexPairSet": [
{
"AppearIndex": 1,
"Index": 0
}
],
"FirstAppear": 1,
"Name": "泡沫板"
},
{
"AppearIndexPairSet": [
{
"AppearIndex": 1,
"Index": 1
}
],
"FirstAppear": 1,
"Name": "宿舍楼"
}
],
"Name": "关键词"
},
{
"AppearIndexPairSet": [],
"FirstAppear": 0,
"L3TagSet": [
{
```

```
"AppearIndexPairSet": [
{
"AppearIndex": 1,
"Index": 2
}
],
"FirstAppear": 1,
"Name": "职工"
}
],
"Name": "身份职位"
},
{
"AppearIndexPairSet": [],
"FirstAppear": 0,
"L3TagSet": [
{
"AppearIndexPairSet": [
{
"AppearIndex": 1,
"Index": 3
}
],
"FirstAppear": 1,
"Name": "济南"
}
],
"Name": "地点"
}
],
"Name": ""
}
]
},
"TitleSet": [],
"WebMediaURL": "https://ai-media-251202827.cos.ap-guangzhou.myqcloud.com/%E6%98%9F%E8%A7%86%E9%A2%91.mp4",
"UnknownPersonSet": [],
"MultiLevelPersonInfoSet": []
}
},
"TaskInfo": {
"FailedReason": "",
"MediaId": "media-tqakf1vs",
"MediaName": "星视频",
"MediaPreknownInfo": {
"MediaLabel": 3,
"MediaLang": 1,
"MediaSecondLabel": 0,
"MediaType": 2
},
"TaskCreateTime": "2021-11-04T12:14:17+08:00",
"TaskId": "task-l0ae7t96",
"TaskName": "",
```

```
"TaskProgress": 0,
"TaskStartTime": "2021-11-04T12:14:18+08:00",
"TaskStatus": 8,
"TaskTimeCost": 110,
"CallbackURL": "http://example.com/api/callback",
"Label": "",
"TextMetadata": null,
"ImageMetadata": null,
"AudioMetadata": null,
"Metadata": null
}
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
FailedOperation.TaskNotFinished	视频分析未完成。
InternalError.DBOperationError	内部DB操作错误。
InternalError.InnerError	内部错误。
InvalidParameter.InvalidTaskId	任务ID不合法。

错误码	描述
ResourceNotFound.RecordNotFound	记录不存在。
ResourceNotFound.TaskNotFound	任务不存在。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

删除任务

最近更新时间：2024-12-04 01:22:29

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

删除任务信息

请注意，本接口**不会**删除媒资文件

只有已完成(成功或者失败)的任务可以删除，**正在执行中的任务不支持删除**

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DeleteTask。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
TaskId	是	String	任务Id 示例值：task-g8lqk737

3. 输出参数

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

4. 示例

示例1 成功删除任务

输入示例

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

{
```

```
"TaskId": "task-kRB7j9e6"
}
```

输出示例

```
{
"Response": {
"RequestId": "5d04776d-f95c-4c7e-be5b-f3bd8ece7219"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
FailedOperation.TaskNotFinished	视频分析未完成。
InvalidParameter.InvalidTaskId	任务ID不合法。
ResourceNotFound.RecordNotFound	记录不存在。
ResourceNotFound.TaskNotFound	任务不存在。
UnsupportedOperation.TaskNotAccessible	任务不可访问。

自定义人物库相关接口
创建自定义人物库

最近更新时间：2024-03-12 01:36:44

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

创建自定义人物库

Bucket的格式参考为 `bucketName-123456.cos.ap-shanghai.myqcloud.com`

在调用CreateCustomPerson和AddCustomPersonImage接口之前，请先确保本接口成功调用。当前每个用户只支持一个自定义人物库，一旦自定义人物库创建成功，后续接口调用均会返回人物库已存在错误。

由于人脸图片对于自定义人物识别至关重要，因此自定义人物识别功能需要用户显式指定COS存储桶方可使用。具体来说，自定义人物识别功能接口(主要是CreateCustomPerson和AddCustomPersonImage)会在此COS桶下面新建IVLDCustomPersonImage目录，并在此目录下存储自定义人物图片数据以支持后续潜在的特征更新。

请注意：本接口指定的COS桶仅用于**备份存储自定义人物图片**，CreateCustomPerson和AddCustomPersonImage接口入参URL可使用任意COS存储桶下的任意图片。

重要：请务必确保本接口指定的COS存储桶存在(不要手动删除COS桶)。COS存储桶一旦指定，将不能修改。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：CreateCustomGroup。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
Bucket	是	String	人脸图片COS存储桶Host地址 示例值：bk0-1234.cos.ap-shanghai.myqcloud.com

3. 输出参数

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

4. 示例

示例1 成功创建自定义人物库

输入示例

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

{
"Bucket": "test-251202813.cos.ap-guangzhou.myqcloud.com"
}
```

输出示例

```
{
"Response": {
"RequestId": "bb0c9a0f-7cb2-4df7-a91f-fbb43eeec80c"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。

错误码	描述
FailedOperation.CustomGroupAlreadyExist	自定义人物库已存在。
InternalError.DBConnectionError	内部DB连接失败。
InternalError.InnerError	内部错误。
InvalidParameter.InvalidURL	URL不合法。
InvalidParameter.UnsupportURL	不支持的URL类型。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

描述自定义人物库

最近更新时间：2024-12-04 01:22:28

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

描述自定义人物库信息，当前库大小(库中有多少人脸)，以及库中的存储桶

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeCustomGroup。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。

3. 输出参数

参数名称	类型	描述
GroupSize	Integer	自定义人物库所包含的人物个数 示例值：100
Bucket	String	自定义人物库图片后续所在的存储桶 示例值：test-251202813.cos.ap-guangzhou.myqcloud.com
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId ）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 成功描述自定义人物库

输入示例

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

{}
```

输出示例

```
{
"Response": {
"Bucket": "test-251202813.cos.ap-guangzhou.myqcloud.com",
"GroupSize": 0,
"RequestId": "dc5b7d40-2210-4298-8809-085a43f64e60"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
InternalError.DBConnectionError	内部DB连接失败。
InternalError.DBOperationError	内部DB操作错误。
InternalError.InnerError	内部错误。
ResourceNotFound.CustomGroupNotFound	自定义人物库不存在。
ResourceNotFound.RecordNotFound	记录不存在。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

创建自定义人物分类

最近更新时间：2024-12-04 01:22:28

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

创建自定义人物分类信息

当L2Category为空时，将创建一级自定义分类。
当L1Category与L2Category均不为空时，将创建二级自定义分类。请注意，**只有当一级自定义分类存在时，才可创建二级自定义分类**。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：CreateCustomCategory。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
L1Category	是	String	自定义一级类型 示例值：娱乐明星
L2Category	是	String	自定义二级类型 示例值：张三

3. 输出参数

参数名称	类型	描述
CategoryId	String	自定义分类信息ID 示例值："10"
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 创建自定义人物分类

输入示例

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

```
<公共请求参数>

{
"L1Category": "公众人物",
"L2Category": "知名医生"
}
```

输出示例

```
{
"Response": {
"CategoryId": "10",
"RequestId": "dffc5c7b-b742-4075-a9a6-9c41d51d6337"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
FailedOperation.CategoryExist	自定义人物分类已存在。
InternalError.DBConnectionError	内部DB连接失败。

错误码	描述
InternalError.InnerError	内部错误。
InvalidParameter.InvalidL1Category	一级自定义类型不合法。
InvalidParameter.InvalidL2Category	二级自定义类型不合法。
InvalidParameter.InvalidParam	输入字段不合法。
ResourceNotFound.CustomCategoryNotFound	自定义人物类型不存在。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

批量描述自定义人物分类

最近更新时间：2024-04-03 11:22:12

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

批量描述自定义人物分类信息

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeCustomCategories。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。

3. 输出参数

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

4. 示例

示例1 描述自定义人物类型

输入示例

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

{}
```

输出示例

```
{
"Response": {
"CategorySet": [
{
"CategoryId": "13",
"L1Category": "公众人物",
"L2Category": "知名律师"
}
],
"RequestId": "2759b810-ad36-4936-93b4-7b374a878e6a"
}
}
```

示例2 未创建任何分类，返回空数组

输入示例

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

{}
```

输出示例

```
{
"Response": {
"CategorySet": [],
"RequestId": "d8461ee7-c4eb-429e-8609-3fa727e5f064"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee

- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
InternalError.DBConnectionError	内部DB连接失败。
InternalError.InnerError	内部错误。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

更新自定义人物分类

最近更新时间：2024-12-04 01:22:27

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

更新自定义人物分类

当L2Category为空时，代表更新CategoryId对应的一级自定义人物类型以及所有二级自定义人物类型所从属的一级自定义人物类型；
当L2Category非空时，仅更新CategoryId对应的二级自定义人物类型

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：UpdateCustomCategory。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
CategoryId	是	String	自定义人物类型Id 示例值："2"
L1Category	是	String	一级自定义人物类型 示例值：公众人物
L2Category	否	String	二级自定义人物类型 示例值：娱乐明星

3. 输出参数

参数名称	类型	描述
CategoryId	String	成功更新的自定义人物类型Id 示例值："2"
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 成功更新自定义人物类型

输入示例

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

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

{
"L1Category": "公众人物",
"L2Category": "娱乐明星",
"CategoryId": "2"
}
```

输出示例

```
{
"Response": {
"CategoryId": "2",
"RequestId": "0f3d2270-90f4-4d5d-9670-3bd44ac8efa2"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。

错误码	描述
FailedOperation.CategoryExist	自定义人物分类已存在。
FailedOperation.CategoryLevelChanged	自定义类型层级变化。
InternalError.DBConnectionError	内部DB连接失败。
InternalError.InnerError	内部错误。
InternalError.InternalOverflow	自定义人物请求超过限制。
InvalidParameter.InvalidCategoryId	自定义人物类型ID不合法。
InvalidParameter.InvalidL1Category	一级自定义类型不合法。
InvalidParameter.InvalidL2Category	二级自定义类型不合法。
InvalidParameter.InvalidParam	输入字段不合法。
ResourceNotFound.CustomCategoryNotFound	自定义人物类型不存在。
ResourceNotFound.RecordNotFound	记录不存在。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

删除自定义分类

最近更新时间：2024-12-10 01:41:33

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

删除自定义分类信息

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DeleteCustomCategory。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
CategoryId	是	String	自定义分类ID 示例值："99"

3. 输出参数

参数名称	类型	描述
CategoryId	String	自定义分类ID 示例值："99"
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 成功删除自定义人物分类

输入示例

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

{
```

```
"CategoryId": "99"
}
```

输出示例

```
{
"Response": {
"CategoryId": "99",
"RequestId": "94efe9e4-1381-43b4-992b-f3f731ff3387"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
FailedOperation.CategoryReferred	自定义人物分类被引用，不能删除。
InternalError.DBConnectionError	内部DB连接失败。
InternalError.InnerError	内部错误。
InvalidParameter.InvalidCategoryId	自定义人物类型ID不合法。
ResourceNotFound.CustomCategoryNotFound	自定义人物类型不存在。

错误码	描述
ResourceNotFound.RecordNotFound	记录不存在。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

创建自定义人物

最近更新时间：2024-12-10 01:41:33

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

创建自定义人物。

输入人物名称，基本信息，分类信息与人脸图片，创建自定义人物

人脸图片可使用图片数据(base64编码的图片数据)或者图片URL(推荐使用COS以减少下载时间，其他地址也支持)，原始图片优先，也即如果同时指定了图片数据和图片URL，接口将仅使用图片数据

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：CreateCustomPerson。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
Name	是	String	自定义人物姓名 示例值：张三
BasicInfo	是	String	自定义人物简要信息(仅用于标记，不支持检索) 示例值：一线娱乐明星
CategoryId	是	String	自定义分类ID，如不存在接口会报错 示例值："11"
ImageURL	否	String	自定义人物图片URL，可支持任意地址，推荐使用COS 示例值：http://b0-123.cos.ap-shanghai.myqcloud.com/hello.jpg
Image	否	String	原始图片base64编码后的数据 示例值：Qk0WWgsAAADYAoAAA...AtQEAADcCAAABABPD

3. 输出参数

参数名称	类型	描述
PersonId	String	自定义人物Id 示例值：person-Axgo3FYc
ImageInfo	PersonImageInfo	自定义人脸信息
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 成功创建自定义人物

输入示例

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

{
"ImageURL": "https://test-251202813.cos.ap-guangzhou.myqcloud.com/%E6%B2%88%E8%85%BE.jpeg",
"BasicInfo": "测试基本信息",
"Name": "测试姓名",
"CategoryId": "11"
}
```

输出示例

```
{
"Response": {
"ImageInfo": {
"ErrorCode": "OK",
"ErrorMsg": "OK",
"ImageId": "image-Axgo3FYc-egMN",
"ImageURL": "https://test-251202813.cos.ap-guangzhou.myqcloud.com/CustomImages/image-Axgo3FYc-egMN.jpeg"
},
"PersonId": "person-Axgo3FYc",
"RequestId": "705f6aed-f46b-4f43-aeef-31167885d938"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.InvalidSecretId	SecredId失效。
AuthFailure.SecretIdNotFound	SecretId不存在。
AuthFailure.SignatureExpire	签名已过期。
AuthFailure.SignatureFailure	签名校验失败。
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
FailedOperation.DownloadFailed	媒资文件下载失败。
FailedOperation.FeatureAlgoFailed	图片特征提取失败。
FailedOperation.MultipleFacesInImage	图片中包含多张人脸。
FailedOperation.NoFaceInImage	图片中不包含人脸。
FailedOperation.PersonDuplicated	人脸库中存在相似的人脸。
FailedOperation.PersonNumExceeded	自定义人物数量过多。
FailedOperation.QualityAlgoFailed	图片质量分检测失败。
FailedOperation.QualityTooLow	图片质量分过低。
FailedOperation.UploadFailed	上传文件失败。
InternalError.DBConnectionError	内部DB连接失败。
InternalError.DBOperationError	内部DB操作错误。
InternalError.InnerError	内部错误。
InternalError.InternalOverflow	自定义人物请求超过限制。
InvalidParameter.InvalidCategoryId	自定义人物类型ID不合法。
InvalidParameter.InvalidImage	图片不合法。
InvalidParameter.InvalidName	名称不合法。
InvalidParameter.InvalidParam	输入字段不合法。
InvalidParameter.InvalidURL	URL不合法。
InvalidParameter.UnsupportURL	不支持的URL类型。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

描述自定义人物详细信息

最近更新时间：2024-12-04 01:22:27

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

描述自定义人物详细信息，包括人物信息与人物信息

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeCustomPersonDetail。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
PersonId	是	String	自定义人物Id 示例值：person-Axgo3FYc

3. 输出参数

参数名称	类型	描述
PersonInfo	CustomPersonInfo	自定义人物信息
TaskIdSet	Array of String	出现该自定义人物的所有分析人物Id 注意：此字段可能返回 null，表示取不到有效值。 示例值：["task-g8lqk737"]
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 成功描述人物详细信息(包括出现这个人物的任务ID列表)

输入示例

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

{
```

```
"PersonId": "person-Axgo3FYc"
}
```

输出示例

```
{
"Response": {
"PersonInfo": {
"BasicInfo": "测试基本信息",
"CreateTime": "2022-01-25 10:35:47",
"ImageInfoSet": [
{
"ErrorCode": "OK",
"ErrorMsg": "OK",
"ImageId": "image-Axgo3FYc-egMN",
"ImageURL": "https://test-251202813.cos.ap-guangzhou.myqcloud.com/CustomImages/image-Axgo3FYc-egMN.jpeg"
}
],
"L1Category": "明星",
"L2Category": "巨星",
"Name": "测试姓名",
"PersonId": "person-Axgo3FYc"
},
"RequestId": "db04f25b-0d68-4cba-9171-254abc8ff3ac",
"TaskIdSet": [
"task-g8lqk737"
]
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
InternalError.DBConnectionError	内部DB连接失败。
InternalError.InnerError	内部错误。
InvalidParameter.InvalidPersonId	人物ID不合法。
ResourceNotFound.RecordNotFound	记录不存在。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

批量描述自定义人物

最近更新时间：2024-12-16 01:41:48

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

批量描述自定义人物

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeCustomPersons。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
PageNumber	是	Integer	分页序号，从1开始 示例值：1
PageSize	是	Integer	分页数据行数，最多50 示例值：50
SortBy	否	SortBy	排序信息，默认倒序
Filter	否	CustomPersonFilter	自定义人物过滤条件

3. 输出参数

参数名称	类型	描述
TotalCount	Integer	满足过滤条件的自定义人物数量 示例值：798
PersonInfoSet	Array of CustomPersonInfo	自定义人物信息
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 批量描述自定义人物

输入示例

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

{
"PageNumber": "1",
"PageSize": "15"
}
```

输出示例

```
{
"Response": {
"PersonInfoSet": [],
"RequestId": "802f88a4-3c8f-4aff-8328-bc7591fdacc1",
"TotalCount": 0
}
}
```

示例2 成功批量描述人物信息

输入示例

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

{
"PageNumber": "1",
"PageSize": "50"
}
```

输出示例

```
{
"Response": {
"PersonInfoSet": [
{
"BasicInfo": "测试基本信息",
"CreateTime": "2022-01-25 10:35:47",
"ImageInfoSet": [
{
"ErrorCode": "OK",
"ErrorMsg": "OK",
"ImageId": "image-Axgo3FYc-egMN",
"ImageURL": "https://test-251202813.cos.ap-guangzhou.myqcloud.com/CustomImages/image-Axgo3FYc-egMN.jpeg"
```

```
}
],
"L1Category": "明星",
"L2Category": "巨星",
"Name": "测试姓名",
"PersonId": "person-Axgo3FYc"
}
],
"RequestId": "9f5bc011-fa0b-4ad9-a4a1-2481343e9428",
"TotalCount": 1
}
}
```


5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
InternalError.DBConnectionError	内部DB连接失败。
InternalError.InnerError	内部错误。
InvalidParameter.InvalidCategoryId	自定义人物类型ID不合法。
InvalidParameter.InvalidL1Category	一级自定义类型不合法。

错误码	描述
InvalidParameter.InvalidName	名称不合法。
InvalidParameter.InvalidPageNumber	分页序号不合法。
InvalidParameter.InvalidPageSize	分页大小不合法。
InvalidParameter.InvalidParam	输入字段不合法。
InvalidParameter.InvalidPersonId	人物ID不合法。
InvalidParameter.InvalidSortBy	排序字段不合法。
ResourceNotFound.CustomCategoryNotFound	自定义人物类型不存在。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

更新自定义人物信息

最近更新时间：2024-12-04 01:22:27

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

更新自定义人物信息，包括姓名，简要信息，分类信息等

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：UpdateCustomPerson。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
PersonId	是	String	待更新的自定义人物Id 示例值：person-Axgo3FYc
Name	否	String	更新后的自定义人物名称，如为空则不更新 示例值：李四
BasicInfo	否	String	更新后的自定义人物简介，如为空则不更新 示例值：喜剧明星
CategoryId	否	String	更新后的分类信息，如为空则不更新 示例值：10

3. 输出参数

参数名称	类型	描述
PersonId	String	成功更新的自定义人物Id 示例值：person-Axgo3FYc
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 更新人物姓名与基本信息

输入示例

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

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

{
"PersonId": "person-Axgo3FYc",
"BasicInfo": "新测试基本信息",
"Name": "测试姓名B"
}
```

输出示例

```
{
"Response": {
"PersonId": "person-Axgo3FYc",
"RequestId": "78b360f1-fe7a-4576-8ab4-d48d06d44572"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。

错误码	描述
InternalError.DBConnectionError	内部DB连接失败。
InternalError.InnerError	内部错误。
InvalidParameter.InvalidCategoryId	自定义人物类型ID不合法。
InvalidParameter.InvalidName	名称不合法。
InvalidParameter.InvalidParam	输入字段不合法。
InvalidParameter.InvalidPersonId	人物ID不合法。
ResourceNotFound.RecordNotFound	记录不存在。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

删除自定义人物

最近更新时间：2024-12-04 01:22:28

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

删除自定义人物

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DeleteCustomPerson。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
PersonId	是	String	待删除的自定义人物ID 示例值：person-Axgo3FYc

3. 输出参数

参数名称	类型	描述
PersonId	String	已删除的自定义人物Id 示例值：person-Axgo3FYc
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 删除自定义人物

输入示例

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

{
```

```
"PersonId": "person-Axgo3FYc"
}
```

输出示例

```
{
"Response": {
"PersonId": "person-Axgo3FYc",
"RequestId": "c3679728-cd86-4bd9-a829-579e9e885c7c"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
InternalError.DBConnectionError	内部DB连接失败。
InternalError.InnerError	内部错误。
InvalidParameter.InvalidPersonId	人物ID不合法。
ResourceNotFound.RecordNotFound	记录不存在。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

增加自定义人脸图片

最近更新时间：2024-12-10 01:41:33

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

增加自定义人脸图片，每个自定义人物最多可包含10张人脸图片

请注意，与创建自定义人物一样，图片数据优先级优于图片URL优先级

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：AddCustomPersonImage。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
PersonId	是	String	自定义人物Id 示例值：person-Axgo3FYc
ImageURL	否	String	自定义人物图片地址 示例值：https://example-123456.cos.ap-guangzho.myqcloud.com/example.jpg
Image	否	String	图片数据base64之后的结果 示例值：Qk0WWgsAAADYAoAAA...AtQEAADcCAAABABPD

3. 输出参数

参数名称	类型	描述
PersonId	String	自定义人物Id 示例值：person-Axgo3FYc
ImageInfo	PersonImageInfo	自定义人脸图片信息
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 新增自定义人物图片

输入示例

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

{
```

```
"PersonId": "person-Axgo3FYc",
"ImageURL": "https://test-251202813.cos.ap-guangzhou.myqcloud.com/%E6%B2%88%E8%85%BE.jpeg"
}
```

输出示例

```
{
"Response": {
"ImageInfo": {
"ErrorCode": "OK",
"ErrorMsg": "OK",
"ImageId": "image-Axgo3FYc-GOeV",
"ImageURL": "https://test-251202813.cos.ap-guangzhou.myqcloud.com/CustomImages/image-Axgo3FYc-GOeV.jpeg"
},
"PersonId": "person-Axgo3FYc",
"RequestId": "f7efda76-2d0d-487d-9e8c-4c198555cc1f"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。

错误码	描述
FailedOperation.DownloadFailed	媒资文件下载失败。
FailedOperation.FeatureAlgoFailed	图片特征提取失败。
FailedOperation.ImageNumExceeded	图片数量过多。
FailedOperation.MultipleFacesInImage	图片中包含多张人脸。
FailedOperation.NoFaceInImage	图片中不包含人脸。
FailedOperation.PersonNotMatched	人脸图片不属于已知人物。
FailedOperation.QualityAlgoFailed	图片质量分检测失败。
FailedOperation.QualityTooLow	图片质量分过低。
InternalError.DBConnectionError	内部DB连接失败。
InternalError.InnerError	内部错误。
InternalError.InternalOverflow	自定义人物请求超过限制。
InvalidParameter.InvalidImage	图片不合法。
InvalidParameter.InvalidPersonId	人物ID不合法。
InvalidParameter.InvalidURL	URL不合法。
ResourceNotFound.RecordNotFound	记录不存在。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

删除自定义人脸

最近更新时间：2024-12-04 01:22:28

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

删除自定义人脸数据

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DeleteCustomPersonImage。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
PersonId	是	String	自定义人物Id 示例值：person-Axgo3FYc
ImageId	是	String	自定义人脸图片Id 示例值：image-Axgo3FYc-GOeV

3. 输出参数

参数名称	类型	描述
PersonId	String	自定义人物Id 示例值：person-Axgo3FYc
ImageId	String	已删除的人物图片Id 示例值：image-Axgo3FYc-GOeV
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 删除自定义人物图片

输入示例

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

```
<公共请求参数>

{
"PersonId": "person-Axgo3FYc",
"ImageId": "image-Axgo3FYc-GOeV"
}
```

输出示例

```
{
"Response": {
"ImageId": "image-Axgo3FYc-GOeV",
"PersonId": "person-Axgo3FYc",
"RequestId": "88ca11c5-3ab8-4069-b2b9-723f6143d6aa"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
FailedOperation.ImageNumExceeded	图片数量过多。

错误码	描述
InternalError.DBConnectionError	内部DB连接失败。
InternalError.InnerError	内部错误。
InternalError.InternalOverflow	自定义人物请求超过限制。
InvalidParameter.InvalidImageId	图片ID不合法。
InvalidParameter.InvalidPersonId	人物ID不合法。
ResourceNotFound.RecordNotFound	记录不存在。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

创建默认自定义人物类型

最近更新时间：2024-03-12 01:36:44

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

创建默认自定义人物类型

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：CreateDefaultCategories。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。

3. 输出参数

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

4. 示例

示例1 创建默认自定义人物分类信息

输入示例

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

{}
```

输出示例

```
{
"Response": {
"RequestId": "400ceb0d-ab84-4108-99df-9ce07c56522b"
```

```
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
InternalError	内部错误。
InternalError.DBConnectionError	内部DB连接失败。
InternalError.InnerError	内部错误。
UnauthorizedOperation	未授权操作。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

用户管理相关接口
编辑回调地址

最近更新时间：2024-12-04 01:22:29

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

用户设置对应事件的回调地址

回调事件消息通知协议

网络协议

- 回调接口协议目前仅支持http/https协议；
- 请求：HTTP POST 请求，包体内容为 JSON，每一种消息的具体包体内容参见后文。
- 应答：HTTP STATUS CODE = 200，服务端忽略应答包具体内容，为了协议友好，建议客户应答内容携带 JSON： `{"code":0}`

通知可靠性

事件通知服务具备重试能力，事件通知失败后会总计重试3次；
为了避免重试对您的服务器以及网络带宽造成冲击，请保持正常回包。触发重试条件如下：

- 长时间（5 秒）未回包应答。
- 应答 HTTP STATUS 不为200。

回调接口协议

分析任务完成消息回调

参数名称	必选	类型	描述
EventType	是	int	回调时间类型，1-任务分析完成，2-媒资导入完成
TaskId	是	String	任务ID
TaskStatus	是	TaskStatus	任务执行状态
FailedReason	是	String	若任务失败，该字段为失败原因

导入媒资完成消息回调

参数名称	必选	类型	描述
EventType	是	int	回调时间类型，1-任务分析完成，2-媒资导入完成
MediaId	是	String	媒资ID
MediaStatus	是	MediaStatus	媒资导入状态
FailedReason	是	String	若任务失败，该字段为失败原因

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：ModifyCallback。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。
TaskFinishNotifyURL	是	String	任务分析完成后回调地址 示例值：http://example.com/api/task/callback
MediaFinishNotifyURL	是	String	媒体导入完成后回调地址 示例值：http://example.com/api/media/callback

3. 输出参数

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

4. 示例

示例1 成功设置回调地址

输入示例

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

{
"TaskFinishNotifyURL": "https://callback.com",
"MediaFinishNotifyURL": "https://callback.com"
}
```

输出示例

```
{
"Response": {
"RequestId": "705f6aed-f46b-4f43-aeef-31167885d938"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
InternalError	内部错误。
InternalError.DBConnectionError	内部DB连接失败。
InternalError.InnerError	内部错误。
InvalidParameter	参数错误。
InvalidParameter.InvalidParam	输入字段不合法。
InvalidParameter.InvalidURL	URL不合法。
InvalidParameter.NameTooLong	名称超过长度限制。
InvalidParameter.ParamTooLong	参数超过长度限制。
InvalidParameter.URLNotResolved	输入URL域名无法解析。
InvalidParameter.UnsupportURL	不支持的URL类型。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

查询回调设置

最近更新时间：2024-12-04 01:22:29

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

查询用户回调设置

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：QueryCallback。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	是	String	公共参数，详见产品支持的 地域列表。

3. 输出参数

参数名称	类型	描述
TaskFinishNotifyURL	String	任务分析完成后回调地址 示例值：http://example.com/api/task/callback
MediaFinishNotifyURL	String	媒体导入完成后回调地址 示例值：http://example.com/api/media/callback
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 成功查询用户回调地址

输入示例

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

{}
```

输出示例

```
{
"Response": {
"RequestId": "705f6aed-f46b-4f43-aeef-31167885d938",
"TaskFinishNotifyURL": "http://example.com/api/task/callback",
"MediaFinishNotifyURL": "http://example.com/api/media/callback"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
FailedOperation	操作失败。
InternalError	内部错误。
InternalError.DBConnectionError	内部DB连接失败。
InternalError.InnerError	内部错误。
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。

视频缩编相关接口
描述视频缩编任务与任务结果

最近更新时间：2024-12-04 01:22:27

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

描述任务信息，如果任务成功完成，还将返回任务结果

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeVideoSummaryDetail。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	否	String	公共参数，本接口不需要传递此参数。
TaskId	是	String	要查询的任务Id 示例值：task-g8lqk737

3. 输出参数

参数名称	类型	描述
Status	Integer	任务的状态 1: 等待处理中 2: 处理中 3: 处理成功 4: 处理失败 示例值：1
FailedReason	String	如果处理失败，返回失败的原因 示例值：“视频下载失败”
AsrSet	Array of AsrResult	提取出的视频的 Asr 结果
TextSummary	String	文本摘要结果 示例值："文本摘要"
TextSegSet	Array of String	文本摘要分割结果 示例值：["文本摘要分割1", "文本摘要分割2"]
ShotSegSet	Array of ShotInfo	镜头分割结果
TextSegMatchShotScoreSet	Array of TextSegMatchShotScore	数组第 i 个结构 TextSegMatchShotConfidenceSet[i] 表示第 i 个文本摘要分割结果和所有镜头的匹配度。

参数名称	类型	描述
TTSResultURLSet	Array of String	TTS 输出音频下载地址列表 示例值：["https://example-123456.cos.ap-guangzho.myqcloud.com/exmaple.wav"]
VideoResultURL	String	合成视频输出下载地址 示例值：https://example-123456.cos.ap-guangzho.myqcloud.com/exmaple.mp4
VideoRotateResultURL	String	合成后的视频横竖屏转换后的视频下载地址 示例值：https://example-123456.cos.ap-guangzho.myqcloud.com/exmaple_rotate.mp4
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 描述任务信息

描述任务信息，如果任务成功完成，将返回任务结果

输入示例

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

{
"TaskId": "task-5bMQVrmJyPJ"
}
```

输出示例

```
{
"Response": {
"AsrSet": [
{
"Content": "来继续关注工人体育场保护性改造复建工程，八月五号工体保护性改造复建工程正式进入到现场施工阶段，改造后，工人体育场将由综合性体育场转变为一座具有国际一流水准的专业足球场，那改造工程有哪些特点呢？改造之后的宫体会有哪些具体的变化和提升呢？工体的改造又会对我国的足球发展带来哪些新的气象呢？针对相关问题，我们北京广播电视台对六位业内专家进行了专访，那今天播出新闻访谈第六集，二零二三年亚洲杯筹办办公室常务副秘书长何喜走进了我们的访谈间。",
"EndTimeStamp": 38.28,
"StartTimeStamp": 0
},
{
"Content": "二零二三亚洲杯呢，是我们在二零一九年的六月四号啊，通过申办取得了举办前，因为呢在二零零四年的时候，我们中国曾经举办过第十三届亚洲杯，当时呢，北京和其他四个城市举办一些非常成功的亚洲杯，我记得当时我们创造的场均上座啊，是三点二万人。",
"EndTimeStamp": 63.08,
"StartTimeStamp": 39.6
},
{
```

"Content": "是亚洲杯到现在的记录还没被打破，所以亚洲年呢，也希望这届比赛啊，把开幕式、揭幕战以及决赛和闭幕式都放在北京，和其他几个城市一起，伴着最高水准的亚洲杯和喜介绍，第十八届亚洲杯足球赛是在包括北京、上海、天津、重庆等中国十座城市举行，目前，这十座城市的专业足球场地建设已全面铺开，现在呢，我们了解。",
"EndTimeStamp": 91.32,
"StartTimeStamp": 63.08
},
{
"Content": "除了北京的工人体育场是改造福建和天津的泰达足球场是这个呃，附件和这个维修改造以外，与其他的八座体育场都是专业的新建的足球场。",
"EndTimeStamp": 104.46,
"StartTimeStamp": 91.32
},
{
"Content": "嗯，其中呢呃，因为上海和成都的两座足球场在我们二零一八年就已经开工，有望在明年初就完全这个竣工，作为本届亚洲杯最重要的赛场，工体的附件将会引领国内专业足球场的建设，并成为这十座专业球场的标杆，为举办圆满精彩、安全、有序的亚洲杯提供良好的硬件基础，能够听见运动员呼吸，看见运动员的表情，甚至我们的这个碗型结构，让你的球迷互动欢呼才能更好的效果，这十座新建或附件的专业球场都将会成为二零二三年亚洲杯重要的赛事遗产，加速建设专业球场的最终目的还是为了促进更多人喜欢足球，爱上足球，我能不能这么理解您的意思，就是说我学习好坏。",
"EndTimeStamp": 153.92,
"StartTimeStamp": 104.46
},
{
"Content": "和我在什么样条件的教室里学习没有关系，但是如果教师的条件越好，是可以给大家创造能够踏踏实实学习的环境。",
"EndTimeStamp": 163.92,
"StartTimeStamp": 153.92
},
{
"Content": "是这样的一个逻辑吧，可以这么说，那我想呢，我们这个通过办亚洲杯，通过专业球场这种氛围，带动更多的青少年来喜欢足球，来从事足球，当然提供更好的专业的训练条件，让他们能够得到更好的训练保障，我想这个都是一个非常好的硬件条件，营造足球氛围，除了新建和复建专业的足球场地外，还要建立完善的青训体系和教练员体系，这也是随着场馆的建设带给我国足球事业发展的新变化，构建每个城市的青训体系，每年从去年申办开始要培养一百米以上的教练员，就基层的教练员，要构建我们自己的每个青少年比赛的体系，从中学大学这个年龄层次以及地域层次，同时呢还要求每个城市对我们青少年的社会的培养，俱乐部也给他一定的扶持的政策，能够培养一些优秀的后备力量出来，这十座专业球场，除北京的工人体育场和天津泰达足球场外上海。",
"EndTimeStamp": 224.3,
"StartTimeStamp": 163.92
},
{
"Content": "浦东足球场和成都凤山体育中心足球场将于今年底和明年出竣工，其余体育场均会在二零二二年十二月份完工，相信这十座新建和复建的专业足球场地也会成为更多青年人成长的记忆，北京台报道。",
"EndTimeStamp": 240.92,
"StartTimeStamp": 224.3
},
{
"Content": "今天是七夕七夕节，电影方面也有多部爱情或女性题材的新片将要上映，那他们和已经上映的战争巨制八佰、魔法精灵等片共同构成了这个档期的主力。",
"EndTimeStamp": 254.82,
"StartTimeStamp": 241.58
},
{
"Content": "每年七夕节必然少不了爱情主题的影片，今年的七夕节最受关注的是国产奇幻爱情电影，我在时间尽头等你，该片由姚婷婷执导，讲述了男主角李鸿其饰演的林格，为换回李一桐饰演的恋人邱倩的生命付出一切，而当他们再次相遇，邱倩却失去了有关林格的所有记忆的故事，不少观众被影片奇幻设定吸引，也希望能拥有重启时空的机会弥补遗憾，找回错过的爱情，影片的预售票房已突破一千万，可见期待值颇高。",
"EndTimeStamp": 291.42,
"StartTimeStamp": 255.9

```
},
{
"Content": "另一部爱情题材的国产片荞麦疯长，讲述的是三个年轻人追逐梦想的故事，云桥想要活成一部电影里麦有看得见希望的未来，吴峰也要在平凡的日子里突破自己，每个主角都有对梦想的渴求，对美好生活的无限向往，他们因此开启了异乡漂泊之路，该片最大的看点是马思纯，钟楚曦，黄景瑜三位人气青年演员的表现，此外，七夕还有一部进口新片小妇人上映，众所周知，该片改编自同名世界名著，主演阵容相当强大，包括西尔莎罗南，艾玛沃森，梅丽尔，斯特里普等众多知名演员，讲述了美国南北战争时期的四位姐妹寻找真爱，正是新生活时由女孩逐渐成长为女人的过程，该片获得了第九十二届奥斯卡金像奖最佳服装设计奖，配角劳拉邓恩也凭此片获得美国国家影评人协会奖，最佳女配角奖，也因此不少女性观众对这部影片。",
"EndTimeStamp": 352.3,
"StartTimeStamp": 291.82
},
{
"Content": "比较期待北京台报道。",
"EndTimeStamp": 355.2,
"StartTimeStamp": 352.3
}
],
"FailedReason": "",
"RequestId": "40ec2397-e2da-49c5-89a6-f41a54f48b7f",
"ShotSegSet": [
{
"EndTimeStamp": 39.32,
"StartTimeStamp": 0
},
{
"EndTimeStamp": 46.72,
"StartTimeStamp": 39.36
},
{
"EndTimeStamp": 51.72,
"StartTimeStamp": 46.76
},
{
"EndTimeStamp": 55,
"StartTimeStamp": 51.76
},
{
"EndTimeStamp": 57.84,
"StartTimeStamp": 55.04
},
{
"EndTimeStamp": 73.64,
"StartTimeStamp": 57.88
},
{
"EndTimeStamp": 75.28,
"StartTimeStamp": 73.68
},
{
"EndTimeStamp": 77.36,
"StartTimeStamp": 75.32
},
{
"EndTimeStamp": 78.12,
```

```
"StartTimeStamp": 77.4
},
{
"EndTimeStamp": 78.8,
"StartTimeStamp": 78.16
},
{
"EndTimeStamp": 80.72,
"StartTimeStamp": 78.84
},
{
"EndTimeStamp": 83.36,
"StartTimeStamp": 80.76
},
{
"EndTimeStamp": 87.12,
"StartTimeStamp": 83.4
},
{
"EndTimeStamp": 89.2,
"StartTimeStamp": 87.16
},
{
"EndTimeStamp": 102.68,
"StartTimeStamp": 89.24
},
{
"EndTimeStamp": 106.24,
"StartTimeStamp": 102.72
},
{
"EndTimeStamp": 115.84,
"StartTimeStamp": 106.28
},
{
"EndTimeStamp": 118.24,
"StartTimeStamp": 115.88
},
{
"EndTimeStamp": 121.72,
"StartTimeStamp": 118.28
},
{
"EndTimeStamp": 129.28,
"StartTimeStamp": 121.76
},
{
"EndTimeStamp": 133.36,
"StartTimeStamp": 129.32
},
{
"EndTimeStamp": 136.8,
"StartTimeStamp": 133.4
```

```
},
{
"EndTimeStamp": 139.08,
"StartTimeStamp": 136.84
},
{
"EndTimeStamp": 140.96,
"StartTimeStamp": 139.12
},
{
"EndTimeStamp": 141.64,
"StartTimeStamp": 141
},
{
"EndTimeStamp": 145.04,
"StartTimeStamp": 141.68
},
{
"EndTimeStamp": 146.96,
"StartTimeStamp": 145.08
},
{
"EndTimeStamp": 149.72,
"StartTimeStamp": 147
},
{
"EndTimeStamp": 164.88,
"StartTimeStamp": 149.76
},
{
"EndTimeStamp": 170.24,
"StartTimeStamp": 164.92
},
{
"EndTimeStamp": 173.2,
"StartTimeStamp": 170.28
},
{
"EndTimeStamp": 181.8,
"StartTimeStamp": 173.24
},
{
"EndTimeStamp": 183.44,
"StartTimeStamp": 181.84
},
{
"EndTimeStamp": 192.2,
"StartTimeStamp": 183.48
},
{
"EndTimeStamp": 195.4,
"StartTimeStamp": 192.24
},
```

```
{
"EndTimeStamp": 204.76,
"StartTimeStamp": 195.44
},
{
"EndTimeStamp": 207.12,
"StartTimeStamp": 204.8
},
{
"EndTimeStamp": 216.56,
"StartTimeStamp": 207.16
},
{
"EndTimeStamp": 222.12,
"StartTimeStamp": 216.6
},
{
"EndTimeStamp": 231.08,
"StartTimeStamp": 222.16
},
{
"EndTimeStamp": 241.36,
"StartTimeStamp": 231.12
},
{
"EndTimeStamp": 255.72,
"StartTimeStamp": 241.4
},
{
"EndTimeStamp": 257.24,
"StartTimeStamp": 255.76
},
{
"EndTimeStamp": 258.56,
"StartTimeStamp": 257.28
},
{
"EndTimeStamp": 259.84,
"StartTimeStamp": 258.6
},
{
"EndTimeStamp": 260.44,
"StartTimeStamp": 259.88
},
{
"EndTimeStamp": 261.8,
"StartTimeStamp": 260.48
},
{
"EndTimeStamp": 263.4,
"StartTimeStamp": 261.84
},
{
```

```
"EndTimeStamp": 265.64,
"StartTimeStamp": 263.44
},
{
"EndTimeStamp": 266.08,
"StartTimeStamp": 265.68
},
{
"EndTimeStamp": 267.92,
"StartTimeStamp": 266.12
},
{
"EndTimeStamp": 269.36,
"StartTimeStamp": 267.96
},
{
"EndTimeStamp": 270.84,
"StartTimeStamp": 269.4
},
{
"EndTimeStamp": 271.76,
"StartTimeStamp": 270.88
},
{
"EndTimeStamp": 273.12,
"StartTimeStamp": 271.8
},
{
"EndTimeStamp": 274.04,
"StartTimeStamp": 273.16
},
{
"EndTimeStamp": 274.76,
"StartTimeStamp": 274.08
},
{
"EndTimeStamp": 276.4,
"StartTimeStamp": 274.8
},
{
"EndTimeStamp": 276.84,
"StartTimeStamp": 276.44
},
{
"EndTimeStamp": 278.52,
"StartTimeStamp": 276.88
},
{
"EndTimeStamp": 279.48,
"StartTimeStamp": 278.56
},
{
"EndTimeStamp": 281.84,
```

```
"StartTimeStamp": 279.52
},
{
"EndTimeStamp": 282.24,
"StartTimeStamp": 281.88
},
{
"EndTimeStamp": 283.76,
"StartTimeStamp": 282.28
},
{
"EndTimeStamp": 284.4,
"StartTimeStamp": 283.8
},
{
"EndTimeStamp": 285.16,
"StartTimeStamp": 284.44
},
{
"EndTimeStamp": 285.76,
"StartTimeStamp": 285.2
},
{
"EndTimeStamp": 286.48,
"StartTimeStamp": 285.8
},
{
"EndTimeStamp": 287.2,
"StartTimeStamp": 286.52
},
{
"EndTimeStamp": 288.04,
"StartTimeStamp": 287.24
},
{
"EndTimeStamp": 289.36,
"StartTimeStamp": 288.08
},
{
"EndTimeStamp": 289.6,
"StartTimeStamp": 289.4
},
{
"EndTimeStamp": 290.96,
"StartTimeStamp": 289.64
},
{
"EndTimeStamp": 291.68,
"StartTimeStamp": 291
},
{
"EndTimeStamp": 292,
"StartTimeStamp": 291.72
```

```
},
{
"EndTimeStamp": 293.84,
"StartTimeStamp": 292.04
},
{
"EndTimeStamp": 295.16,
"StartTimeStamp": 293.88
},
{
"EndTimeStamp": 295.72,
"StartTimeStamp": 295.2
},
{
"EndTimeStamp": 296.4,
"StartTimeStamp": 295.76
},
{
"EndTimeStamp": 297.2,
"StartTimeStamp": 296.44
},
{
"EndTimeStamp": 298.44,
"StartTimeStamp": 297.24
},
{
"EndTimeStamp": 300.48,
"StartTimeStamp": 298.48
},
{
"EndTimeStamp": 302.16,
"StartTimeStamp": 300.52
},
{
"EndTimeStamp": 303.44,
"StartTimeStamp": 302.2
},
{
"EndTimeStamp": 305.16,
"StartTimeStamp": 303.48
},
{
"EndTimeStamp": 306.16,
"StartTimeStamp": 305.2
},
{
"EndTimeStamp": 307.2,
"StartTimeStamp": 306.2
},
{
"EndTimeStamp": 308.36,
"StartTimeStamp": 307.24
},
```

```
{
"EndTimeStamp": 308.6,
"StartTimeStamp": 308.4
},
{
"EndTimeStamp": 309.96,
"StartTimeStamp": 308.64
},
{
"EndTimeStamp": 310.6,
"StartTimeStamp": 310
},
{
"EndTimeStamp": 311.6,
"StartTimeStamp": 310.64
},
{
"EndTimeStamp": 312.68,
"StartTimeStamp": 311.64
},
{
"EndTimeStamp": 313.24,
"StartTimeStamp": 312.72
},
{
"EndTimeStamp": 314.6,
"StartTimeStamp": 313.28
},
{
"EndTimeStamp": 315.44,
"StartTimeStamp": 314.64
},
{
"EndTimeStamp": 316.04,
"StartTimeStamp": 315.48
},
{
"EndTimeStamp": 317.56,
"StartTimeStamp": 316.08
},
{
"EndTimeStamp": 318.44,
"StartTimeStamp": 317.6
},
{
"EndTimeStamp": 319.16,
"StartTimeStamp": 318.48
},
{
"EndTimeStamp": 319.88,
"StartTimeStamp": 319.2
},
{
```

```
"EndTimeStamp": 320.52,
"StartTimeStamp": 319.92
},
{
"EndTimeStamp": 321.88,
"StartTimeStamp": 320.56
},
{
"EndTimeStamp": 323.16,
"StartTimeStamp": 321.92
},
{
"EndTimeStamp": 324.08,
"StartTimeStamp": 323.2
},
{
"EndTimeStamp": 325.68,
"StartTimeStamp": 324.12
},
{
"EndTimeStamp": 327.28,
"StartTimeStamp": 325.72
},
{
"EndTimeStamp": 329.88,
"StartTimeStamp": 327.32
},
{
"EndTimeStamp": 330.76,
"StartTimeStamp": 329.92
},
{
"EndTimeStamp": 331.68,
"StartTimeStamp": 330.8
},
{
"EndTimeStamp": 333.2,
"StartTimeStamp": 331.72
},
{
"EndTimeStamp": 333.88,
"StartTimeStamp": 333.24
},
{
"EndTimeStamp": 334.76,
"StartTimeStamp": 333.92
},
{
"EndTimeStamp": 335.6,
"StartTimeStamp": 334.8
},
{
"EndTimeStamp": 337.32,
```

```
"StartTimeStamp": 335.64
},
{
"EndTimeStamp": 338.6,
"StartTimeStamp": 337.36
},
{
"EndTimeStamp": 339.44,
"StartTimeStamp": 338.64
},
{
"EndTimeStamp": 340.2,
"StartTimeStamp": 339.48
},
{
"EndTimeStamp": 341.4,
"StartTimeStamp": 340.24
},
{
"EndTimeStamp": 342.48,
"StartTimeStamp": 341.44
},
{
"EndTimeStamp": 343,
"StartTimeStamp": 342.52
},
{
"EndTimeStamp": 345.04,
"StartTimeStamp": 343.04
},
{
"EndTimeStamp": 345.36,
"StartTimeStamp": 345.08
},
{
"EndTimeStamp": 346.2,
"StartTimeStamp": 345.4
},
{
"EndTimeStamp": 346.68,
"StartTimeStamp": 346.24
},
{
"EndTimeStamp": 347.16,
"StartTimeStamp": 346.72
},
{
"EndTimeStamp": 348,
"StartTimeStamp": 347.2
},
{
"EndTimeStamp": 348.72,
"StartTimeStamp": 348.04
```

```
},
{
"EndTimeStamp": 349.72,
"StartTimeStamp": 348.76
},
{
"EndTimeStamp": 350.52,
"StartTimeStamp": 349.76
},
{
"EndTimeStamp": 351.2,
"StartTimeStamp": 350.56
},
{
"EndTimeStamp": 352.24,
"StartTimeStamp": 351.24
},
{
"EndTimeStamp": 352.52,
"StartTimeStamp": 352.28
},
{
"EndTimeStamp": 353.24,
"StartTimeStamp": 352.56
},
{
"EndTimeStamp": 353.44,
"StartTimeStamp": 353.28
},
{
"EndTimeStamp": 353.76,
"StartTimeStamp": 353.48
},
{
"EndTimeStamp": 355.28,
"StartTimeStamp": 353.8
}
],
"Status": 3,
"TTSResultURLSet": [],
"TextSegMatchShotScoreSet": [
{
"ScoreSet": [
1.2382,
0.2471,
0.3201,
0.276,
0.2811,
0.2292,
0.2716,
0.2528,
0.335,
0.2754,
```

```
0.2779,
0.3048,
0.3227,
0.3249,
0.2848,
0.2373,
0.2443,
0.295,
0.319,
0.3246,
0.2584,
0.3243,
0.2845,
0.2877,
0.2871,
0.3045,
0.281,
0.2661,
0.2205,
0.2509,
0.2244,
0.248,
0.2875,
0.3396,
0.3332,
0.2238,
0.2127,
0.2448,
0.3186,
0.385,
0.3784,
0.2365,
0.1854,
0.1944,
0.164,
0.1984,
0.1925,
0.1828,
0.1792,
0.19,
0.1965,
0.1944,
0.1925,
0.2063,
0.189,
0.1935,
0.1879,
0.195,
0.1648,
0.1506,
0.1923,
0.2021,
0.1986,
```

```
0.1832,
0.1625,
0.1885,
0.187,
0.183,
0.1923,
0.1956,
0.1737,
0.1702,
0.1578,
0.1947,
0.1969,
0.1877,
0.20077057,
0.1817064,
0.18227036,
0.16899092,
0.20706016,
0.17507376,
0.1820092,
0.18835092,
0.17191781,
0.16255495,
0.15076947,
0.18277371,
0.16016904,
0.16486597,
0.15029283,
0.1576431,
0.18024676,
0.17325334,
0.16157725,
0.16709505,
0.14414482,
0.1432873,
0.17011009,
0.14700972,
0.17108813,
0.14706013,
0.14398573,
0.13889273,
0.14886412,
0.14262123,
0.13954282,
0.12645954,
0.13878693,
0.12918936,
0.118234135,
0.12329925,
0.1154507,
0.11960802,
0.10515184,
0.13556333,
```

```
0.13373142,
0.13648494,
0.118176214,
0.13478275,
0.11370157,
0.10680597,
0.10862925,
0.101744354,
0.11934858,
0.11186232,
0.10477673,
0.098862946,
0.13034238,
0.09909322,
0.118106075,
0.102340594,
0.10789193,
0.108175315,
0.1057501,
0.104622856,
0.0923
]
},
{
"ScoreSet": [
0.09792612,
0.10299725,
0.15805012,
0.11373982,
0.13031949,
0.10293696,
0.12196275,
0.13122137,
0.16946042,
0.1464477,
0.1390185,
0.15382768,
0.17648277,
0.168605,
1.1611,
1.1084,
0.122020416,
0.17283109,
1.1165,
0.17653316,
0.14276025,
0.1679703,
0.15868554,
0.15540288,
0.1569139,
0.15847902,
0.1691859,
0.16199781,
```

```
0.1442033,
0.1491408,
0.15405715,
0.14157155,
0.18572833,
1.1075,
0.18936706,
0.14374636,
0.1542158,
0.15442917,
1.0741,
1.0415,
0.25049958,
0.15842898,
0.15485612,
0.1537589,
0.13934197,
0.15227734,
0.1543463,
0.15192968,
0.14826345,
0.15826666,
0.16319416,
0.17251374,
0.17194976,
0.16985875,
0.17011373,
0.1756132,
0.16527703,
0.17321062,
0.15579715,
0.1410848,
0.1843216,
0.1895185,
0.17970891,
0.17051928,
0.14806718,
0.17878558,
0.16995937,
0.1711359,
0.19963719,
0.1832138,
0.16902883,
0.16724464,
0.15241763,
0.18540592,
0.1876833,
0.17974508,
0.2025312,
0.1852,
0.1865,
0.1754,
0.2007,
```

```
0.1816,
0.1929,
0.2139,
0.1831,
0.1568,
0.1691,
0.2005,
0.1786,
0.1943,
0.167,
0.1827,
0.1831,
0.1892,
0.1825,
0.2036,
0.1776,
0.166,
0.2137,
0.1639,
0.2189,
0.1705,
0.1853,
0.16,
0.19296151,
0.17859381,
0.1791361,
0.17200361,
0.1656221,
0.16876182,
0.1451389,
0.12888582,
0.13670826,
0.12979229,
0.13301837,
0.14549954,
0.15055135,
0.14475827,
0.1273419,
0.14195904,
0.12808834,
0.1245674,
0.10474258,
0.113638274,
0.124841385,
0.11840741,
0.11249482,
0.119784206,
0.12193935,
0.09766378,
0.10952644,
0.10293357,
0.105447516,
0.0981708,
```

```
0.09860591,
0.09366528,
0.08725
]
},
{
"ScoreSet": [
0.10146389,
0.09110581,
0.13973857,
0.11085404,
0.12418423,
0.09422003,
0.13293672,
0.121666946,
0.16040991,
0.12902275,
0.13101485,
0.14889637,
0.1540935,
0.16147652,
1.1381,
1.0963,
0.10722985,
0.15270078,
1.1043,
0.1901058,
0.12855065,
0.17453684,
0.13663934,
0.13847779,
0.13567795,
0.1558032,
0.15463743,
0.14819604,
0.11811904,
0.12851611,
0.13397743,
1.0504,
0.17586869,
1.0416,
0.20025352,
0.124881364,
0.12813942,
0.12954707,
0.19087067,
0.21714239,
0.22443168,
0.1455727,
0.103213094,
0.10952031,
0.1028835,
0.112136126,
```

```
0.11812128,
0.10620863,
0.11484153,
0.111630835,
0.12100338,
0.13205242,
0.13011758,
0.13772815,
0.11843302,
0.12385813,
0.12259317,
0.13591461,
0.10702,
0.10594032,
0.14232999,
0.13548459,
0.13760749,
0.12786406,
0.12088954,
0.13425413,
0.1333774,
0.13110718,
0.15116458,
0.14035,
0.121839926,
0.11671858,
0.11092556,
0.13467526,
0.14016877,
0.120735355,
0.1319307,
0.12738395,
0.11492863,
0.115515664,
0.15610728,
0.118331365,
0.12884498,
0.14165401,
0.12537254,
0.12588383,
0.11667716,
0.15081502,
0.12062214,
0.13688324,
0.12223233,
0.13832806,
0.14421274,
0.13880803,
0.14029342,
0.14342754,
0.13143517,
0.12227128,
0.1693569,
```

```
0.14405952,
0.17145987,
0.13747562,
0.13964434,
0.15077971,
0.15429644,
0.1252,
0.1402,
0.1333,
0.1372,
0.1427,
0.1217,
0.1304,
0.1313,
0.1381,
0.1272,
0.1677,
0.1583,
0.1697,
0.1587,
0.1683,
0.1355,
0.1426,
0.1442,
0.1431,
0.1576,
0.139,
0.129,
0.142,
0.1791,
0.1424,
0.1606,
0.1302,
0.1504,
0.1616,
0.15,
0.1491,
0.06945
]
}
],
"TextSegSet": [
"国际一流体育场改造后的工体将由综合性体育场转变为一座具有国际一流水准的专业足球场，改造后的整体会有哪些具体变化和提升，专家称改造后的工体会有哪些具体的变化和提升",
"除了北京的工人体育场是改，造福建和天津的泰达足球场是这个",
"附件和这个维修改造以外，与其他的八座体育场都是专业的新建的足球场"
],
"TextSummary": "国际一流体育场改造后的工体将由综合性体育场转变为一座具有国际一流水准的专业足球场，改造后的整体会有哪些具体变化和提升，专家称改造后的工体会有哪些具体的变化和提升。除了北京的工人体育场是改，造福建和天津的泰达足球场是这个，附件和这个维修改造以外，与其他的八座体育场都是专业的新建的足球场。",
"VideoResultURL": "",
"VideoRotateResultURL": ""
```

```
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

视频摘要使用量统计

最近更新时间：2024-07-09 14:27:27

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

获取用户资源使用量

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeUsageAmount。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	否	String	公共参数，本接口不需要传递此参数。

3. 输出参数

参数名称	类型	描述
UsedHours	Float	资源使用小时数 示例值：1.5
TotalHours	Float	资源包总量小时数 示例值：1000
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。 定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 获取资源使用量

获取资源使用量

输入示例

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

{}
```

输出示例

```
{
"Response": {
"RequestId": "f2f1a66b-5c63-46dc-93c3-5132f10cb713",
"TotalHours": 2,
"UsedHours": 0.593375
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

创建视频缩编任务

最近更新时间：2024-12-04 01:22:27

1. 接口描述

接口请求域名： ivld.tencentcloudapi.com 。

创建一个视频缩编任务。

回调事件消息通知协议

网络协议

- 回调接口协议目前仅支持http/https协议；
- 请求：HTTP POST 请求，包体内容为 JSON，每一种消息的具体包体内容参见后文。
- 应答：HTTP STATUS CODE = 200，服务端忽略应答包具体内容，为了协议友好，建议客户应答内容携带 JSON： `{"code":0}`

通知可靠性

事件通知服务具备重试能力，事件通知失败后会总计重试3次；
为了避免重试对您的服务器以及网络带宽造成冲击，请保持正常回包。触发重试条件如下：

- 长时间（5 秒）未回包应答。
- 应答 HTTP STATUS 不为200。

回调接口协议

分析任务完成消息回调

参数名称	必选	类型	描述
TaskId	是	String	任务ID
TaskStatus	是	Integer	任务执行状态
FailedReason	是	String	若任务失败，该字段为失败原因

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

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

2. 输入参数

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

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：CreateVideoSummaryTask。
Version	是	String	公共参数，本接口取值：2021-09-03。
Region	否	String	公共参数，本接口不需要传递此参数。
SummaryType	是	Integer	目前只支持 1，表示新闻缩编。 示例值：1

参数名称	必选	类型	描述
VideoURL	是	String	待处理的视频的URL，目前只支持*不带签名的*COS地址，长度最长1KB 示例值：cos://my-release-9327555674/d833e1e4bb2.mp4
CallbackURL	否	String	任务处理完成的回调地址。 示例值：http://example.com/api/callback
WriteBackCosPath	否	String	如果需要你输出 TTS 或者视频，该字段为转存的cos桶地址且不可为空; 示例：https://${Bucket}-${AppId}.cos.${Region}.myqcloud.com/${PathPrefix}/(注意，cos路径需要以/分隔符结尾)。 示例值：https://my-release-9327555674.cos.ap-guangzhou.myqcloud.com/writedir/
ActiveVideoGenerate	否	Boolean	是否开启结果视频生成功能，如果开启，需要指定WriteBackCosPath 参数 示例值：true
VideoRotationMode	否	VideoRotationMode	生成结果视频的时候，控制生成的结果视频的横转竖参数。如果 ActiveVideoGenerate 为 false, 该参数无效。
TTSMode	否	TTSMode	语音合成相关的控制参数
ActiveTTSOutput	否	Boolean	是否输出合成好的语音列表。 示例值：true
ExactAsrSet.N	否	Array of AsrResult	用户指定的精确的 asr 结果列表
ExactTextSummary	否	String	用户指定的精确的文本摘要 示例值："文本摘要"
ExactTextSegSet.N	否	Array of String	用户指定的精确的文本摘要分割结果 示例值：["文本摘要分割1", "文本摘要分割2"]
ExactShotSegSet.N	否	Array of ShotInfo	用户指定的精确的镜头分割结果

3. 输出参数

参数名称	类型	描述
TaskId	String	返回的任务 id 示例值：task-g8lqk737
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 创建视频缩编任务

创建视频缩编任务，并且生成视频回写到 cos。

输入示例

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

{
"SummaryType": 1,
```

```
"VideoURL": "https://ai-media-1256936300.cos.ap-guangzhou.myqcloud.com/test_data/test-beijingninzao-6mins.mp4",
"WriteBackCosPath": "https://test-media-ai-251202827.cos.ap-guangzhou.myqcloud.com/ai-media/video-summary/",
"ActiveVideoGenerate": true
}
```

输出示例

```
{
"Response": {
"RequestId": "5646aef2-9481-409b-ba6f-91c40f5f9de6",
"TaskId": "task-5bMQVrmJyPJ"
}
}
```

5. 开发者资源

腾讯云 API 平台

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

API Inspector

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

SDK

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

- Tencent Cloud SDK 3.0 for Python: GitHub Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

数据结构

最近更新时间：2024-12-10 01:41:34

AppearIndexPair

出现信息索引对

AppearIndex可选值定义如下：

AppearIndex名称	AppearIndex取值	AppearIndex描述
APPEAR_INDEX_INVALID	0	非法的任务状态
APPEAR_INDEX_AUDIO	1	音频出现信息
APPEAR_INDEX_TEXT	2	可视文本出现信息
APPEAR_INDEX_VIDEO	3	视频出现信息

例如，当AppearIndex=1，Index=15，则意味着目标关键词出现在第16个(Index计数从0开始)音频文字识别结果之中

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
AppearIndex	Integer	出现信息，取值范围为[1，3] 示例值：1
Index	Integer	AppearInfo中AppearIndex对应元素的第Index元素，从0开始计数 示例值：50

AppearInfo

出现信息结构

包含关键词在音频转文字(ASR)，图片转文字(OCR)以及视频结果中的出现信息

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
AudioAppearSet	Array of TextAppearInfo	关键词在音频文本结果中的出现位置数组 注意：此字段可能返回 null，表示取不到有效值。
TextAppearSet	Array of TextAppearInfo	关键词在可视文本结果中的出现位置数组 注意：此字段可能返回 null，表示取不到有效值。
VideoAppearSet	Array of VideoAppearInfo	关键词在视频信息中的出现位置数组 注意：此字段可能返回 null，表示取不到有效值。

AsrResult

一条 asr 语音结果的结构

被如下接口引用：CreateVideoSummaryTask, DescribeVideoSummaryDetail。

名称	类型	必选	描述
Content	String	是	ASR提取的文字信息 示例值："欢迎收看新闻30分"

名称	类型	必选	描述
StartTimeStamp	Float	是	ASR起始时间戳，从0开始 示例值：0.52
EndTimeStamp	Float	是	ASR结束时间戳，从0开始 示例值：1.92

AudioData

音频文件分析结果数据

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
AudioInfoSet	Array of AudioInfo	音频识别文本结果 注意：此字段可能返回 null，表示取不到有效值。
TextTagSet	MultiLevelTag	音频识别标签数据
WebMediaURL	String	音频下载地址 注意：此字段可能返回 null，表示取不到有效值。 示例值：https://example-109235.cos.ap-guangzho.myqcloud.com/example.mp3

AudioInfo

音频识别结果信息

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
Content	String	ASR提取的文字信息 示例值："欢迎收看新闻30分"
StartTimeStamp	Float	ASR起始时间戳，从0开始 示例值：0.52
EndTimeStamp	Float	ASR结束时间戳，从0开始 示例值：0.92
Tag	String	ASR提取的音频标签 示例值："导语"

AudioMetadata

音频文件元信息

被如下接口引用：DescribeMedia, DescribeMedias, DescribeTask, DescribeTaskDetail, DescribeTasks。

名称	类型	描述
FileSize	Integer	媒资音频文件大小，单位为Byte 注意：此字段可能返回 null，表示取不到有效值。 示例值：1024
MD5	String	媒资音频文件MD5 注意：此字段可能返回 null，表示取不到有效值。 示例值：bbc3f6c9fb18d80dedf28239f9a86fac
Duration	Float	媒资音频时长，单位为秒 注意：此字段可能返回 null，表示取不到有效值。

名称	类型	描述
		示例值：20.00
SampleRate	Float	媒资音频采样率，单位为khz 注意：此字段可能返回 null，表示取不到有效值。 示例值：44.1
BitRate	Integer	媒资音频码率，单位为kbps 注意：此字段可能返回 null，表示取不到有效值。 示例值：128
Format	String	媒资音频文件格式 注意：此字段可能返回 null，表示取不到有效值。 示例值：mp3
BitDepth	Integer	Audio Bit Depth: 16/24 bit .etc 注意：此字段可能返回 null，表示取不到有效值。 示例值：24
ShortFormat	String	封装格式短后缀 注意：此字段可能返回 null，表示取不到有效值。 示例值：mp3

ClassifiedPersonInfo

已分类的人物信息

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
ClassifyName	String	人物分类名称 注意：此字段可能返回 null，表示取不到有效值。 示例值：演员
PersonInfoSet	Array of PersonInfo	符合特定分类的人物信息数组 注意：此字段可能返回 null，表示取不到有效值。

CustomCategory

自定义分类信息

被如下接口引用：DescribeCustomCategories。

名称	类型	描述
CategoryId	String	自定义分类ID 示例值："10"
L1Category	String	一级自定义类型 示例值：娱乐明星
L2Category	String	二级自定义类型 示例值：张三

CustomPersonFilter

自定义人物批量查询过滤条件

被如下接口引用：DescribeCustomPersons。

名称	类型	必选	描述
Name	String	是	待查询的人物姓名 示例值：张三
CategoryIdSet	Array of String	是	待过滤的自定义类型Id数组 示例值：["10", "99"]
PersonIdSet	Array of String	是	待过滤的自定义人物Id数组 示例值：["person-Axgo3FYc", "person-Bxgo3FYc"]
L1CategorySet	Array of String	是	一级自定义人物类型数组 示例值：["喜剧明星", "话剧明星"]

CustomPersonInfo

自定义人物信息

被如下接口引用：DescribeCustomPersonDetail, DescribeCustomPersons。

名称	类型	描述
PersonId	String	自定义人物Id 示例值：person-Axgo3FYc
Name	String	自定义人物姓名 示例值：张三
BasicInfo	String	自定义人物简介信息 示例值：著名喜剧明星
L1Category	String	一级自定义人物类型 示例值：娱乐明星
L2Category	String	二级自定义人物类型 示例值：喜剧演员
ImageInfoSet	Array of PersonImageInfo	自定义人物图片信息
CreateTime	String	自定义人物创建时间 示例值：2022-01-25 10:35:47

Data

任务结果数据

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
ShowInfo	ShowInfo	节目粒度结构化结果 注意：此字段可能返回 null，表示取不到有效值。

ImageData

图片文件标签结果

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
OcrSet	Array of ImageOcr	图片中出现的可视文本识别结果 注意：此字段可能返回 null，表示取不到有效值。

名称	类型	描述
FrameTagSet	MultiLevelTag	图片中出现的帧标签识别结果 注意：此字段可能返回 null，表示取不到有效值。
MultiLevelPersonInfoSet	Array of MultiLevelPersonInfo	图片中出现的层级人物识别结果 注意：此字段可能返回 null，表示取不到有效值。
TvLogo	ImageLogo	图片中出现的台标识别结果 注意：此字段可能返回 null，表示取不到有效值。
SourceLogo	ImageLogo	图片中出现的来源信息识别结果 注意：此字段可能返回 null，表示取不到有效值。

ImageLogo

图片中出现的Logo信息

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
Logo	String	图片中出现的Logo识别结果 注意：此字段可能返回 null，表示取不到有效值。 示例值：“山东卫视”
AppearRect	Rectf	Logo在图片中出现的位置 注意：此字段可能返回 null，表示取不到有效值。

ImageMetadata

图片文件元信息

被如下接口引用：DescribeMedia, DescribeMedias, DescribeTask, DescribeTaskDetail, DescribeTasks。

名称	类型	描述
FileSize	Integer	媒资图片文件大小，单位为Byte 注意：此字段可能返回 null，表示取不到有效值。 示例值：1024
MD5	String	媒资图片文件MD5 注意：此字段可能返回 null，表示取不到有效值。 示例值：bbc3f6c9fb18d80dedf28239f9a86fac
Width	Integer	媒资图片文件宽度 注意：此字段可能返回 null，表示取不到有效值。 示例值：1920
Height	Integer	媒资图片文件高度 注意：此字段可能返回 null，表示取不到有效值。 示例值：1080
Format	String	媒资图片文件格式 注意：此字段可能返回 null，表示取不到有效值。 示例值：jpg

ImageOcr

图片OCR识别结果

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
Content	String	图片中可视文本识别结果 注意：此字段可能返回 null，表示取不到有效值。 示例值：“欢迎收看测试新闻”
AppearRect	Rectf	可视文本在图片中的位置信息 注意：此字段可能返回 null，表示取不到有效值。

L1Tag

一级标签信息

请注意，一级标签信息可能不包含二级标签(此时L2TagSet为空)。在这种情况下，一级标签可直接包含出现信息。

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
Name	String	一级标签名 示例值："地点"
L2TagSet	Array of L2Tag	二级标签数组 注意：此字段可能返回 null，表示取不到有效值。
AppearIndexPairSet	Array of AppearIndexPair	一级标签出现信息 注意：此字段可能返回 null，表示取不到有效值。
FirstAppear	Integer	一级标签首次出现信息 注意：此字段可能返回 null，表示取不到有效值。 示例值：1

L2Tag

二级标签信息

请注意，二级标签信息可能不包含三级标签(此时L3TagSet为空)。

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
Name	String	二级标签名 示例值："地点"
L3TagSet	Array of L3Tag	从属于此二级标签的三级标签数组 注意：此字段可能返回 null，表示取不到有效值。
AppearIndexPairSet	Array of AppearIndexPair	二级标签出现信息 注意：此字段可能返回 null，表示取不到有效值。
FirstAppear	Integer	二级标签首次出现信息 注意：此字段可能返回 null，表示取不到有效值。 示例值：1

L3Tag

三级标签信息。

三级标签不再包含任何子标签。所有三级标签都对应着识别结果中的出现信息，出现信息使用AppearIndexPairSet定位。

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
Name	String	三级标签名 示例值："成都"
AppearIndexPairSet	Array of AppearIndexPair	三级标签出现信息索引数组 注意：此字段可能返回 null，表示取不到有效值。
FirstAppear	Integer	三级标签首次出现信息，可选值为[1,3] 注意：此字段可能返回 null，表示取不到有效值。 示例值：1

MediaFilter

媒资过滤条件

被如下接口引用：DescribeMedias。

名称	类型	必选	描述
MediaNameSet	Array of String	否	媒资名称过滤条件 示例值：["新闻30分","朝闻天下"]
StatusSet	Array of Integer	否	媒资状态数组，媒资状态可选值参见MediaInfo 示例值：[1,2]
MediaIdSet	Array of String	否	媒资ID数组 示例值：["59","108"]
LabelSet	Array of String	否	媒资自定义标签数组 示例值：["新闻"]
MediaType	Integer	否	媒资文件类型，定义参见MediaPreknownInfo.MediaType 示例值：2

MediaInfo

媒资信息结构体

媒资状态定义如下：

状态名	状态值	状态描述
MEDIA_STATUS_INVALID	0	非法状态
MEDIA_STATUS_WAITING	1	等待中
MEDIA_STATUS_DOWNLOADING	2	下载中
MEDIA_STATUS_DOWNLOADED	3	下载完成
MEDIA_STATUS_DOWNLOAD_FAILED	4	下载失败(已废弃)
MEDIA_STATUS_TRANSCODING	5	转码中
MEDIA_STATUS_TRANSCODED	6	转码完成
MEDIA_STATUS_TRANCODE_FAILED	7	转码失败(已废弃)
MEDIA_STATUS_SUCCESS	8	媒资文件状态就绪，可发起任务
MEDIA_STATUS_EXPIRED	9	媒资文件已过期
MEDIA_STATUS_FAILED	10	媒资导入失败

被如下接口引用：DescribeMedia, DescribeMedias。

名称	类型	描述
MediaId	String	媒资ID 示例值：“media-2aHsU6sj”
Name	String	媒资名称 注意：此字段可能返回 null，表示取不到有效值。 示例值："新闻30分"
DownLoadURL	String	媒资下载地址 注意：此字段可能返回 null，表示取不到有效值。 示例值："http://example-bucket/example-key.mp4"
Status	Integer	媒资状态，取值参看上方表格 注意：此字段可能返回 null，表示取不到有效值。 示例值：3
FailedReason	String	若状态为失败，表示失败原因 注意：此字段可能返回 null，表示取不到有效值。 示例值："下载失败"
Metadata	MediaMetadata	媒资视频元信息，仅在MediaType=VIDEO时有效 注意：此字段可能返回 null，表示取不到有效值。
Progress	Float	导入视频进度，取值范围为[0,100] 注意：此字段可能返回 null，表示取不到有效值。 示例值：30.0
Label	String	媒资自定义标签 注意：此字段可能返回 null，表示取不到有效值。 示例值："tag"
CallbackURL	String	媒资导入完成后的回调地址 注意：此字段可能返回 null，表示取不到有效值。 示例值："http://example.com/api/callback"
MediaType	Integer	媒资文件类型，具体参看MediaPreknownInfo 注意：此字段可能返回 null，表示取不到有效值。 示例值：1
AudioMetadata	AudioMetadata	媒资音频元信息，仅在MediaType=Audio时有效 注意：此字段可能返回 null，表示取不到有效值。
ImageMetadata	ImageMetadata	媒资图片文件元信息，仅在MediaType=Image时有效 注意：此字段可能返回 null，表示取不到有效值。
TextMetadata	TextMetadata	媒资文本文件元信息，仅在MediaType=Text时有效 注意：此字段可能返回 null，表示取不到有效值。

MediaMetadata

媒资文件视频元信息，包括分辨率，帧率，码率等

被如下接口引用：DescribeMedia, DescribeMedias, DescribeTask, DescribeTaskDetail, DescribeTasks。

名称	类型	描述
FileSize	Integer	媒资视频文件大小，单位为字节 示例值：340000
MD5	String	媒资视频文件MD5 示例值："d228d21b68688c14dcb7275b806398fc"

名称	类型	描述
Duration	Float	媒资视频时长，单位为秒 注意：此字段可能返回 null，表示取不到有效值。 示例值：20.00
NumFrames	Integer	媒资视频总帧数 注意：此字段可能返回 null，表示取不到有效值。 示例值：500
Width	Integer	媒资视频宽度，单位为像素 注意：此字段可能返回 null，表示取不到有效值。 示例值：1920
Height	Integer	媒资视频高度，单位为像素 注意：此字段可能返回 null，表示取不到有效值。 示例值：1080
FPS	Float	媒资视频帧率，单位为Hz 注意：此字段可能返回 null，表示取不到有效值。 示例值：25
BitRate	Integer	媒资视频比特率，单位为kbps 注意：此字段可能返回 null，表示取不到有效值。 示例值：5000

MediaPreknownInfo

描述输入媒资的先验知识，例如文件类型(视频)，媒体类型(新闻/综艺等)

MediaPreknownInfo.MediaType:

MediaType 名称	MediaType取值	MediaType描述
MEDIA_TYPE_INVALID	0	非法的媒资文件类型
MEDIA_TYPE_IMAGE	1	图片
MEDIA_TYPE_VIDEO	2	视频
MEDIA_TYPE_AUDIO	3	音频
MEDIA_TYPE_VIDEO_STREAM	4	视频流，暂不支持
MEDIA_TYPE_TEXT	5	文本

MediaPreknownInfo.MediaLabel:

MediaLabel名称	MediaLabel取值	MediaLabel描述
MEDIA_LABEL_INVALID	0	非法的一级媒资素材类型
MEDIA_LABEL_NEWS	1	新闻
MEDIA_LABEL_ENTERTAINMENT	2	综艺
MEDIA_LABEL_INTERNET_INFO	3	互联网资讯
MEDIA_LABEL_MOVIE	4	电影
MEDIA_LABEL_SERIES	5	电视连续剧
MEDIA_LABEL_SPECIAL	6	专题
MEDIA_LABEL_SPORT	7	体育

MediaPreknownInfo.MediaSecondLabel

请注意：当且仅当MediaLabel=2(综艺)时MediaSecondLabel才有意义

MediaSecondLabel名称	MediaSecondLabel取值	MediaSecondLabel 描述
MEDIA_SECOND_LABEL_INVALID	0	非法的MediaSecondLabel
MEDIA_SECOND_LABEL_EVENING	1	综艺晚会
MEDIA_SECOND_LABEL_OTHERS	2	其他

MediaMeta.MediaLang

MediaLang名称	MediaLang取值	MediaLang描述
MEDIA_LANG_INVALID	0	非法的MediaLang
MEDIA_LANG_MANDARIN	1	普通话
MEDIA_LANG_CANTONESE	2	粤语

被如下接口引用：CreateTask, DescribeTask, DescribeTaskDetail, DescribeTasks。

名称	类型	必选	描述
MediaType	Integer	是	媒资文件类型，参见MediaPreknownInfo结构体定义 示例值：1
MediaLabel	Integer	是	媒资素材一级类型，参见MediaPreknownInfo结构体定义 示例值：1
MediaSecondLabel	Integer	是	媒资素材二级类型，参见MediaPreknownInfo结构体定义 注意：此字段可能返回 null，表示取不到有效值。 示例值：1
MediaLang	Integer	否	媒资音频类型，参见MediaPreknownInfo结构体定义 注意：此字段可能返回 null，表示取不到有效值。 示例值：1

MultiLevelPersonInfo

带类型树的已分类人物信息

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
L1ClassifyName	String	一级分类名称(分类信息参见自定义人物类型) 注意：此字段可能返回 null，表示取不到有效值。 示例值：明星
L2ClassifiedPersonInfoSet	Array of ClassifiedPersonInfo	已分类人物信息数组(所有分类类型为二级分类) 注意：此字段可能返回 null，表示取不到有效值。
Source	Integer	检测结果来源 注意：此字段可能返回 null，表示取不到有效值。 示例值：0

MultiLevelTag

标签信息结构体

包含多级(最多三级)标签结果，以及这些标签在识别结果中的出现位置

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
TagSet	Array of L1Tag	树状标签信息 注意：此字段可能返回 null，表示取不到有效值。
AppearInfo	AppearInfo	标签在识别结果中的定位信息 注意：此字段可能返回 null，表示取不到有效值。

PersonImageInfo

自定义人物人脸图片信息

被如下接口引用：AddCustomPersonImage, CreateCustomPerson, DescribeCustomPersonDetail, DescribeCustomPersons。

名称	类型	描述
ImageId	String	人脸图片ID 示例值："941d96ae-52d9-4401-be6a-fcf1bab6a214"
ImageURL	String	自定义人脸图片的URL，存储在IVLDCustomPreson存储桶内 示例值：https://example-123456.cos.ap-guangzho.myqcloud.com/example.jpg
ErrorCode	String	自定义人脸图片处理错误码 示例值：OK
ErrorMsg	String	自定义人脸图片处理错误信息 示例值：OK

PersonInfo

人物信息

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
Name	String	公众人物姓名 示例值：陈好
Job	String	公众人物职务 示例值：演员
FirstAppear	Integer	首次出现模态，可选值为[1,3]，详细参见AppearIndex定义 示例值：1
AppearInfo	AppearInfo	人物出现信息
AppearRect	Rectf	人脸在图片中的位置，仅在图片标签任务有效 注意：此字段可能返回 null，表示取不到有效值。
PersonId	String	人物的personId 注意：此字段可能返回 null，表示取不到有效值。 示例值：person-Axgo3FYc

Rectf

矩形内容框

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
X	Float	矩形框左上角水平座标 注意：此字段可能返回 null，表示取不到有效值。 示例值：20
Y	Float	矩形框左上角竖直座标 注意：此字段可能返回 null，表示取不到有效值。 示例值：20
Width	Float	矩形框宽度 注意：此字段可能返回 null，表示取不到有效值。 示例值：300
Height	Float	矩形框长度 注意：此字段可能返回 null，表示取不到有效值。 示例值：20

ShotInfo

输入的镜头信息的描述

被如下接口引用：CreateVideoSummaryTask, DescribeVideoSummaryDetail。

名称	类型	必选	描述
StartTimeStamp	Float	是	镜头开始时间 注意：此字段可能返回 null，表示取不到有效值。 示例值：0.52
EndTimeStamp	Float	是	镜头结束时间 注意：此字段可能返回 null，表示取不到有效值。 示例值：1.92

ShowInfo

视频结构化结果

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
Date	String	节目日期(只在新闻有效) 注意：此字段可能返回 null，表示取不到有效值。 示例值：2021-01-01
Logo	String	台标 注意：此字段可能返回 null，表示取不到有效值。 示例值：东方卫视
Column	String	节目名称 注意：此字段可能返回 null，表示取不到有效值。 示例值：东方新闻
Source	String	来源信息 注意：此字段可能返回 null，表示取不到有效值。 示例值：东方卫视
CoverImageURL	String	节目封面 注意：此字段可能返回 null，表示取不到有效值。 示例值：https://example-123456.cos.ap-guangzho.myqcloud.com/example-key

名称	类型	描述
SummarySet	Array of String	节目内容概要列表 注意：此字段可能返回 null，表示取不到有效值。 示例值：["全运会召开",”神舟飞船升空"]
TitleSet	Array of String	节目片段标题列表 注意：此字段可能返回 null，表示取不到有效值。 示例值：["中国共产党第十九届中央委员会第五次全体会议在京召开"]
AudioInfoSet	Array of AudioInfo	音频识别结果列表 注意：此字段可能返回 null，表示取不到有效值。
TextInfoSet	Array of TextInfo	可视文字识别结果列表 注意：此字段可能返回 null，表示取不到有效值。
ClassifiedPersonInfoSet	Array of ClassifiedPersonInfo	已分类人物信息列表 注意：此字段可能返回 null，表示取不到有效值。
TextTagSet	MultiLevelTag	文本标签列表，包含标签内容和出现信息 注意：此字段可能返回 null，表示取不到有效值。
FrameTagSet	MultiLevelTag	帧标签列表，包括人物信息，场景信息等 注意：此字段可能返回 null，表示取不到有效值。
WebMediaURL	String	视频下载地址 注意：此字段可能返回 null，表示取不到有效值。 示例值：https://example-123456.cos.ap-guangzho.myqcloud.com/example-key
MediaClassifierSet	Array of String	媒资分类信息 注意：此字段可能返回 null，表示取不到有效值。 示例值：["会议"]
SummaryTagSet	Array of String	概要标签信息 注意：此字段可能返回 null，表示取不到有效值。 示例值：["全运会"]
UnknownPersonSet	Array of UnknownPerson	未知人物信息 注意：此字段可能返回 null，表示取不到有效值。
MultiLevelPersonInfoSet	Array of MultiLevelPersonInfo	树状已分类人物信息 注意：此字段可能返回 null，表示取不到有效值。

SortBy

排序条件

被如下接口引用：DescribeCustomPersons, DescribeMedias, DescribeTasks。

名称	类型	必选	描述
By	String	否	排序字段，默认为CreateTime 示例值：CreateTime
Descend	Boolean	否	true表示降序，false表示升序 示例值：true

TTSMode

TTS 的参数模式

被如下接口引用：CreateVideoSummaryTask。

名称	类型	必选	描述
Speed	Float	否	语速，范围：[-2，2]，分别对应不同语速： -2代表0.6倍 -1代表0.8倍 0代表1.0倍（默认） 1代表1.2倍 2代表1.5倍 如果需要更细化的语速，可以保留小数点后 2 位，例如0.5/1.25/2.81等。 示例值：1
VoiceType	Integer	否	音色 ID，音色体验地址。 |音乐ID|音色名称|推荐场景| |--|--|--| |1001|智瑜|情感女声| |1002|智聆|通用女声| |1003|智美|客服女声| |1004|智云|通用男声| |1005|智莉|通用女声| |1007|智娜|客服女声| |1008|智琪|客服女声| |1009|智芸|知性女声| |1010|智华|通用男声| |1017|智蓉|情感女声| |1018|智靖|情感男声| 示例值：1001

TaskFilter

任务筛选条件结构体

被如下接口引用：DescribeTasks。

名称	类型	必选	描述
MediaTypeSet	Array of Integer	否	媒资文件类型 示例值：[2]
TaskStatusSet	Array of Integer	否	待筛选的任务状态列表 示例值：[1,2,3]
TaskNameSet	Array of String	否	待筛选的任务名称数组 示例值：["xx新闻","yy新闻"]
TaskIdSet	Array of String	否	TaskId数组 示例值：["task-g8lqk737", "task-b8lqk738"]
MediaNameSet	Array of String	否	媒资文件名数组 示例值：["xx新闻"]
MediaLangSet	Array of Integer	否	媒资语言类型 示例值：[1,2]
MediaLabelSet	Array of Integer	否	媒资素材一级类型 示例值：[1,3]
LabelSet	Array of String	否	媒资自定义标签数组 示例值：["新闻"]

TaskInfo

任务信息

TaskStatus定义如下:

TaskStatus名称	TaskStatus取值	TaskStatus描述
TASK_STATUS_INVALID	0	非法的任务状态
TASK_STATUS_WAITING	1	排队中
TASK_STATUS_ANALYSING	2	任务分析中
TASK_STATUS_ANALYSED	3	任务分析完成
TASK_STATUS_SNAPSHOT_CREATING	4	任务结果快照生成中
TASK_STATUS_SNAPSHOT_CREATED	5	任务结果快照生成完成
TASK_STATUS_RESULT_UPLOADING	6	任务结果快照上传中
TASK_STATUS_RESULT_UPLOADED	7	任务结果快照上传完成
TASK_STATUS_SUCCESS	8	任务执行完成
TASK_STATUS_FAILED	9	任务执行失败

被如下接口引用：DescribeTask, DescribeTaskDetail, DescribeTasks。

名称	类型	描述
TaskId	String	任务ID 示例值：“task-g8lqk737”
TaskName	String	描述任务名称，指定后可根据名称筛选 注意：此字段可能返回 null，表示取不到有效值。 示例值："本地新闻"
MediaId	String	媒资文件ID 示例值："media-2aHsU6sj"
TaskStatus	Integer	任务执行状态 示例值：1
TaskProgress	Float	任务进度，范围为[0，100] 注意：此字段可能返回 null，表示取不到有效值。 示例值：75
TaskTimeCost	Integer	任务执行时间 注意：此字段可能返回 null，表示取不到有效值。 示例值：300
TaskCreateTime	String	任务创建时间 示例值：2006-01-02T15:04:05Z07:00
TaskStartTime	String	任务开始执行时间 注意：此字段可能返回 null，表示取不到有效值。 示例值：2006-01-02T15:04:05Z07:00
FailedReason	String	任务失败原因 注意：此字段可能返回 null，表示取不到有效值。 示例值："分析失败"
MediaPreknownInfo	MediaPreknownInfo	任务执行时指定的先验知识

名称	类型	描述
MediaName	String	媒资文件名称 注意：此字段可能返回 null，表示取不到有效值。 示例值："test-video"
Label	String	媒资自定义标签 注意：此字段可能返回 null，表示取不到有效值。 示例值："tag"
CallbackURL	String	任务分析完成后的后调地址 注意：此字段可能返回 null，表示取不到有效值。 示例值："http://example.com/api/callback"
AudioMetadata	AudioMetadata	任务对应的媒资文件元信息，仅在MediaType为Audio时有效 注意：此字段可能返回 null，表示取不到有效值。
ImageMetadata	ImageMetadata	任务对应的媒资文件元信息，仅在MediaType为Audio时有效 注意：此字段可能返回 null，表示取不到有效值。
TextMetadata	TextMetadata	任务对应的媒资文件元信息，仅在MediaType为Text时有效 注意：此字段可能返回 null，表示取不到有效值。
Metadata	MediaMetadata	任务对应的媒资文件元信息，仅在MediaType为Video时有效 注意：此字段可能返回 null，表示取不到有效值。

TextAppearInfo

关键词在文本中的定位信息

Position为关键词在文本中的偏移量，从0开始。例如，给定文本结果"欢迎收看新闻三十分”，以及关键词"新闻三十分"，那么StartPosition的值为4，EndPosition的值为9

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
Index	Integer	文本结果数组中的下标 示例值：1
StartPosition	Integer	关键词在文本中出现的起始偏移量(包含) 示例值：10
EndPosition	Integer	关键词在文本中出现的结束偏移量(不包含) 示例值：13

TextData

文本文件标签识别结果

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
Content	String	文本内容信息 注意：此字段可能返回 null，表示取不到有效值。 示例值：“欢迎收看测试新闻”
Summary	String	文本概要信息 注意：此字段可能返回 null，表示取不到有效值。 示例值：“欢迎”
TextTagSet	MultiLevelTag	文本标签信息 注意：此字段可能返回 null，表示取不到有效值。

名称	类型	描述
WebMediaURL	String	文档下载地址 注意：此字段可能返回 null，表示取不到有效值。 示例值：https://example-109235.cos.ap-guangzho.myqcloud.com/example.json

TextInfo

可视文本识别结果信息(OCR)

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
Content	String	OCR提取的内容 示例值："深圳湾公园启用"
StartTimeStamp	Float	OCR起始时间戳，从0开始 示例值：0.52
EndTimeStamp	Float	OCR结束时间戳，从0开始 示例值：1.52
Tag	String	OCR标签信息 示例值："新闻"

TextMetadata

文本文件元信息

被如下接口引用：DescribeMedia, DescribeMedias, DescribeTask, DescribeTaskDetail, DescribeTasks。

名称	类型	描述
FileSize	Integer	媒资文本文件大小，单位为字节 注意：此字段可能返回 null，表示取不到有效值。 示例值：1024
MD5	String	媒资文本文件MD5 注意：此字段可能返回 null，表示取不到有效值。 示例值：bbc3f6c9fb18d80dedf28239f9a86fac
Length	Integer	媒资文本文件字符数 注意：此字段可能返回 null，表示取不到有效值。 示例值：30
Format	String	媒资文本文件格式 注意：此字段可能返回 null，表示取不到有效值。 示例值：txt
ShortFormat	String	封装格式短后缀 注意：此字段可能返回 null，表示取不到有效值。 示例值：txt

TextSegMatchShotScore

单个文本摘要分割结果和所有镜头的匹配度信息

被如下接口引用：DescribeVideoSummaryDetail。

名称	类型	描述
ScoreSet	Array of Float	数组第 i 个值表示该文本摘要和第 i 个镜头的匹配度 注意：此字段可能返回 null，表示取不到有效值。 示例值：["0.1", "0.98", "0.65"]

UnknownPerson

未知人物信息

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
VideoAppearSet	Array of VideoAppearInfo	视觉出现信息 注意：此字段可能返回 null，表示取不到有效值。
PutLibraryAllowed	Boolean	未知人物是否可以入库(只有当未知人物人脸小图质量分符合要求时才可入库) 注意：此字段可能返回 null，表示取不到有效值。 示例值：true
AuditClass	Integer	内容审核结果: 0-正常;1-涉政;其他待确定 注意：此字段可能返回 null，表示取不到有效值。 示例值：0

VideoAppearInfo

关键词在视觉结果中的定位信息

被如下接口引用：DescribeTaskDetail。

名称	类型	描述
StartTimeStamp	Float	视觉信息起始时间戳，从0开始 示例值：0.52
EndTimeStamp	Float	视觉信息终止时间戳，从0开始 关键词在视觉信息中的区间为[StartTimeStamp, EndTimeStamp) 示例值：1.52
ImageURL	String	关键词在视觉信息中的封面图片 示例值：cos://my-release-9327555674/xfyheve9rf5.jpg

VideoRotationMode

视频横转竖的控制参数

被如下接口引用：CreateVideoSummaryTask。

名称	类型	必选	描述
ActiveVideoRotation	Boolean	是	生成的视频是否需要横屏转竖屏。 示例值：true

错误码

最近更新时间：2024-12-20 01:37:14

功能说明

如果返回结果中存在 Error 字段，则表示调用 API 接口失败。例如：

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

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

错误码列表

公共错误码

错误码	说明
ActionOffline	接口已下线。
AuthFailure.InvalidAuthorization	请求头部的 `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。

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

业务错误码

错误码	说明
AuthFailure.TaskFinished	任务已完成。
AuthFailure.UserActivated	用户已激活。
AuthFailure.UserInvalidStatus	用户状态异常。
AuthFailure.UserNotFound	用户无权限。
AuthFailure.UserStopArrear	用户已欠费停服。
FailedOperation.AiTemplateNotExist	匹配的模板不存在。
FailedOperation.CategoryExist	自定义人物分类已存在。
FailedOperation.CategoryLevelChanged	自定义类型层级变化。
FailedOperation.CategoryReferred	自定义人物分类被引用，不能删除。
FailedOperation.CustomGroupAlreadyExist	自定义人物库已存在。

错误码	说明
FailedOperation.DBConnectionError	内部DB连接失败。
FailedOperation.DownloadFailed	媒资文件下载失败。
FailedOperation.FeatureAlgoFailed	图片特征提取失败。
FailedOperation.GetCAMTokenFailed	获取CAM临时鉴权失败。
FailedOperation.GetTaskListFailed	获取任务列表失败。
FailedOperation.GetVideoMetadataFailed	获取媒资信息失败。
FailedOperation.ImageNumExceeded	图片数量过多。
FailedOperation.MD5Mismatch	MD5不匹配。
FailedOperation.MediaAlreadyExist	媒资文件已经存在。
FailedOperation.MediaExpired	媒资文件已经过期。
FailedOperation.MediaInUse	媒资正在使用。
FailedOperation.MediaNotReady	媒体文件未就绪。
FailedOperation.MultipleFacesInImage	图片中包含多张人脸。
FailedOperation.NoFaceInImage	图片中不包含人脸。
FailedOperation.OpenChargeFailed	计费开通失败。
FailedOperation.PersonDuplicated	人脸库中存在相似的人脸。
FailedOperation.PersonNotMatched	人脸图片不属于已知人物。
FailedOperation.PersonNumExceeded	自定义人物数量过多。
FailedOperation.QualityAlgoFailed	图片质量分检测失败。
FailedOperation.QualityTooLow	图片质量分过低。
FailedOperation.SnapshotDeserializeFailed	结果快照反序列化失败。
FailedOperation.StopFlowFailed	停止AI工作室任务失败。
FailedOperation.TaskAlreadyExist	存在相同的任务。
FailedOperation.TaskNotFinished	视频分析未完成。
FailedOperation.TranscodeFailed	转码失败。
FailedOperation.UploadFailed	上传文件失败。
InternalError.DBConnectionError	内部DB连接失败。
InternalError.DBOperationError	内部DB操作错误。
InternalError.InnerError	内部错误。
InternalError.InternalOverflow	自定义人物请求超过限制。
InvalidParameter.InvalidCategoryId	自定义人物类型ID不合法。
InvalidParameter.InvalidFilePath	文件路径不合法。
InvalidParameter.InvalidImage	图片不合法。
InvalidParameter.InvalidImageId	图片ID不合法。

错误码	说明
InvalidParameter.InvalidL1Category	一级自定义类型不合法。
InvalidParameter.InvalidL2Category	二级自定义类型不合法。
InvalidParameter.InvalidMD5	MD5不合法。
InvalidParameter.InvalidMediaId	媒体ID不合法。
InvalidParameter.InvalidMediaLabel	MediaLabel无效。
InvalidParameter.InvalidMediaLang	MediaLang无效。
InvalidParameter.InvalidMediaName	媒体名称非法。
InvalidParameter.InvalidMediaPreknownInfo	MediaPreknownInfo无效。
InvalidParameter.InvalidMediaStatus	媒资状态不合法。
InvalidParameter.InvalidMediaType	MediaType无效。
InvalidParameter.InvalidName	名称不合法。
InvalidParameter.InvalidPageNumber	分页序号不合法。
InvalidParameter.InvalidPageSize	分页大小不合法。
InvalidParameter.InvalidParam	输入字段不合法。
InvalidParameter.InvalidPersonId	人物ID不合法。
InvalidParameter.InvalidSortBy	排序字段不合法。
InvalidParameter.InvalidSortOrder	排序方式不合法。
InvalidParameter.InvalidTaskId	任务ID不合法。
InvalidParameter.InvalidTaskName	任务名称不合法。
InvalidParameter.InvalidTaskStatus	任务状态不合法。
InvalidParameter.InvalidURL	URL不合法。
InvalidParameter.InvalidUin	用户Uin无效。
InvalidParameter.NameTooLong	名称超过长度限制。
InvalidParameter.ParamTooLong	参数超过长度限制。
InvalidParameter.URLNotResolved	输入URL域名无法解析。
InvalidParameter.UnsupportURL	不支持的URL类型。
LimitExceeded.UsageLimitExceeded	使用量超过限制。
RequestLimitExceeded.BatchImportOverflow	批量导入超过限制。
RequestLimitExceeded.ConcurrencyOverflow	同时发起过多任务。
ResourceNotFound.CustomCategoryNotFound	自定义人物类型不存在。
ResourceNotFound.CustomGroupNotFound	自定义人物库不存在。
ResourceNotFound.MediaNotFound	媒资文件不存在。
ResourceNotFound.RecordNotFound	记录不存在。
ResourceNotFound.TaskNotFound	任务不存在。

错误码	说明
UnauthorizedOperation.UnauthorizedProduct	用户未激活该产品。
UnsupportedOperation.MediaNotAccessible	媒资文件不可访问。
UnsupportedOperation.TaskNotAccessible	任务不可访问。