

Image Moderation System

API Documentation

Product Documentation



Copyright Notice

©2013-2025 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

API Documentation

- History

- Introduction

- API Category

- Making API Requests

 - Request Structure

 - Common Params

 - Signature v3

 - Signature

 - Responses

- Image Moderation APIs

 - ImageModeration

- Data Types

- Error Codes

API Documentation

History

Last updated : 2024-11-25 17:03:38

Release 2

Release time: 2023-03-24 17:17:01

Release updates:

Improvement to existing documentation.

Modified APIs:

- [ImageModeration](#)
 - New output parameters:RecognitionResults

New data structures:

- [RecognitionResult](#)
- [RecognitionTag](#)

Release 1

Release time: 2022-03-28 10:53:59

Release updates:

Improvement to existing documentation.

New APIs:

- [ImageModeration](#)

New data structures:

- [Device](#)
- [LabelDetailItem](#)
- [LabelResult](#)
- [LibDetail](#)
- [LibResult](#)

- [Location](#)
- [ObjectDetail](#)
- [ObjectResult](#)
- [OcrResult](#)
- [OcrTextDetail](#)
- [User](#)

Introduction

Last updated : 2024-11-25 17:03:29

Image Moderation System (IMS) can accurately recognize content in images that may be offensive, unsafe, or inappropriate. It allows you to configure an image blocklist to recognize custom types of content.

API Category

Last updated : 2024-11-25 17:03:30

Image Moderation APIs

| API Name | Feature | Frequency Limit (maximum requests per second) |
|---------------------------------|-----------------|---|
| ImageModeration | Moderates image | 100 |

Making API Requests

Request Structure

Last updated : 2024-11-25 17:03:30

1. Service Address

The API supports access from either a nearby region (at `ims.intl.tencentcloudapi.com`) or a specified region (at `ims.ap-guangzhou.tencentcloudapi.com` for Guangzhou, for example).

We recommend using the domain name to access the nearest server. When you call an API, the request is automatically resolved to a server in the region **nearest** to the location where the API is initiated. For example, when you initiate an API request in Guangzhou, this domain name is automatically resolved to a Guangzhou server, the result is the same as that of specifying the region in the domain like "`ims.ap-guangzhou.tencentcloudapi.com`".

Note: For latency-sensitive businesses, we recommend that you specify the region in the domain name.

Tencent Cloud currently supports the following regions:

| Hosted region | Domain name |
|--|---|
| Local access region (recommended, only for non-financial availability zones) | <code>ims.intl.tencentcloudapi.com</code> |
| South China (Guangzhou) | <code>ims.ap-guangzhou.tencentcloudapi.com</code> |
| East China (Shanghai) | <code>ims.ap-shanghai.tencentcloudapi.com</code> |
| North China (Beijing) | <code>ims.ap-beijing.tencentcloudapi.com</code> |
| Southwest China (Chengdu) | <code>ims.ap-chengdu.tencentcloudapi.com</code> |
| Southwest China (Chongqing) | <code>ims.ap-chongqing.tencentcloudapi.com</code> |
| Hong Kong, Macao, Taiwan (Hong Kong, China) | <code>ims.ap-hongkong.tencentcloudapi.com</code> |
| Southeast Asia (Singapore) | <code>ims.ap-singapore.tencentcloudapi.com</code> |
| Southeast Asia (Bangkok) | <code>ims.ap-bangkok.tencentcloudapi.com</code> |
| | |

| | |
|----------------------------------|--|
| South Asia (Mumbai) | ims.ap-mumbai.tencentcloudapi.com |
| Northeast Asia (Seoul) | ims.ap-seoul.tencentcloudapi.com |
| Northeast Asia (Tokyo) | ims.ap-tokyo.tencentcloudapi.com |
| U.S. East Coast (Virginia) | ims.na-ashburn.tencentcloudapi.com |
| U.S. West Coast (Silicon Valley) | ims.na-siliconvalley.tencentcloudapi.com |
| Europe (Frankfurt) | ims.eu-frankfurt.tencentcloudapi.com |

2. Communications Protocol

All the Tencent Cloud APIs communicate via HTTPS, providing highly secure communication tunnels.

3. Request Methods

Supported HTTP request methods:

- POST (recommended)
- GET

The Content-Type types supported by POST requests:

- application/json (recommended). The TC3-HMAC-SHA256 signature algorithm must be used.
- application/x-www-form-urlencoded. The HmacSHA1 or HmacSHA256 signature algorithm must be used.
- multipart/form-data (only supported by certain APIs). You must use TC3-HMAC-SHA256 to calculate the signature.

The size of a GET request packet is up to 32 KB. The size of a POST request is up to 1 MB when the HmacSHA1 or HmacSHA256 signature algorithm is used, and up to 10 MB when TC3-HMAC-SHA256 is used.

4. Character Encoding

Only UTF-8 encoding is used.

Common Params

Last updated : 2024-11-25 17:03:31

Common parameters are used for all APIs authenticating requestors. Common parameters must be included in all API requests, and they will not be described in individual API documents.

The exact contents of the common parameters will vary depending on the version of the signature method you use.

Common parameters for Signature Algorithm v3

When the TC3-HMAC-SHA256 algorithm is used, the common parameters should be uniformly placed in the HTTP request header, as shown below:

| Parameter Name | Type | Required | Description |
|----------------|---------|----------|---|
| X-TC-Action | String | Yes | The name of the API for the desired operation. For the specific value, see description of common parameter <code>Action</code> in the input parameters in related API documentation. For example, the API for querying the CVM instance list is <code>DescribeInstances</code> . |
| X-TC-Region | String | Yes | Region parameter, which is used to identify the region to which the data you work with belongs. For values supported for an API, see the description of common parameter <code>Region</code> in the input parameters in related API documentation. This parameter is not required for some APIs (which will be indicated in related API documentation), and will not take effect even it is passed. |
| X-TC-Timestamp | Integer | Yes | The current UNIX timestamp that records the time when the API request is sent. For example, 1529223702. Note: If the difference between the UNIX timestamp and server time is greater than 5 minutes, a signature expiration error may occur. |
| X-TC-Version | String | Yes | API version of the action. For the valid values, see the description of the common parameter <code>Version</code> in the API documentation. For example, the version is 2017-03-12. |
| Authorization | String | Yes | The HTTP authentication request header, for example: TC3-HMAC-SHA256 Credential=AKID*****/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc96317 Here: - TC3-HMAC-SHA256: Signature method, currently fixed as this value; - Credential: Signature credential; AKID***** is the SecretId; Date is a date and time, and this value must match the value of X-TC-Timestamp (a common parameter). |

| | | | |
|------------|--------|----|--|
| | | | <p>UTC time format; service is the name of the product/service, and is general name prefix. For example, a domain name cvm.tencentcloudapi.com refers to product and the value would be cvm;</p> <ul style="list-style-type: none"> - SignedHeaders: The headers that contains the authentication information type and host are the required headers; - Signature: Signature digest. |
| X-TC-Token | String | No | <p>The token used for a temporary certificate. It must be used with a temporary key. You can obtain the temporary key and token by calling a CAM API. No token is required for a long-term key.</p> |

Assuming you want to query the list of Cloud Virtual Machine instances in the Guangzhou region, the request structure in the form of request URL, request header and request body may be as follows:

Example of an HTTP GET request structure:

```

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

Authorization: TC3-HMAC-SHA256 Credential=AKID*****/2018-10-09/cvm/tc3_request, SignedHeaders=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
    
```

The following example shows you how to structure an HTTP POST (application/json) request:

```

https://cvm.tencentcloudapi.com/

Authorization: TC3-HMAC-SHA256 Credential=AKID*****/2018-05-30/cvm/tc3_request, SignedHeaders=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}
    
```

Example of an HTTP POST (multipart/form-data) request structure (only supported by specific APIs):

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

```
Authorization: TC3-HMAC-SHA256 Credential=AKID*****/2018-05-30/cvm/tc3_request, SignedHeaders=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--
```

Common parameters for Signature Algorithm v1

To adopt the HmacSHA1 and HmacSHA256 signature methods, common parameters must be put into the request string, as shown below:

| Parameter Name | Type | Required | Description |
|----------------|--------|----------|--|
| Action | String | Yes | The name of the API for the desired operation. For the specific value, see the description of common parameter <code>Action</code> in the input parameters in related API documentation. For example, the API for querying the CVM instance list is <code>DescribeInstances</code> . |
| Region | String | Yes | Region parameter, which is used to identify the region to which the data you want to work with belongs. For values supported for an API, see the description of common parameter <code>Region</code> in the input parameters in related API documentation. Note: This parameter is not required for some APIs (which will be indicated in related API documentation), and will not take effect even if it is passed. |

| | | | |
|-----------------|---------|-----|---|
| Timestamp | Integer | Yes | The current UNIX timestamp that records the time when the API request was initiated, for example, 1529223702. If the difference between the value and the current system time is too large, a signature expiration error may occur. |
| Nonce | Integer | Yes | A random positive integer used along with <code>Timestamp</code> to prevent replay attacks. |
| SecretId | String | Yes | The identifying SecretId obtained on the Cloud API Key page. A SecretId corresponds to a unique SecretKey which is used to generate the request signature (Signature). |
| Signature | String | Yes | Request signature used to verify the validity of this request. This is calculated based on the actual input parameters. For more information about how this is calculated, see the API authentication documentation. |
| Version | String | Yes | API version of the action. For the valid values, see the description of the common input parameter <code>Version</code> in the API documentation. For example, the version of CVM is 2017-03-12. |
| SignatureMethod | String | No | Signature method. Currently, only HmacSHA256 and HmacSHA1 are supported. The HmacSHA256 algorithm is used to verify the signature only when this parameter is specified as HmacSHA256. In other cases, the signature is verified with HmacSHA1. |
| Token | String | No | The token used for a temporary certificate. It must be used with a temporary key. You can obtain the temporary key and token by calling a CAM API. No token is required for a long-term key. |

Assuming you want to query the list of Cloud Virtual Machine instances in the Guangzhou region, the request structure in the form of request URL, request header and request body may be as follows:

Example of an HTTP GET request structure:

```
https://cvm.tencentcloudapi.com/?Action=DescribeInstances&Version=2017-03-12&SignatureMethod=HmacSHA256&Timestamp=1527672334&Signature=37ac2f4fde00b0ac9bd9eadeb459b1bbbee224158d66e7ae5fcadb70b2d181d02&Region=ap-guangzhou&Nonce=23823223&SecretId=AKID*****
```

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

Example of an HTTP POST request structure:

```
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*****  
*****
```

Region List

The supported Region field values for all APIs in this product are listed as below. For any API that does not support any of the following regions, this field will be described additionally in the relevant API document.

| Region | Value |
|----------------------------|------------------|
| North China (Beijing) | ap-beijing |
| South China (Guangzhou) | ap-guangzhou |
| South Asia (Mumbai) | ap-mumbai |
| East China (Shanghai) | ap-shanghai |
| Southeast Asia (Singapore) | ap-singapore |
| Europe (Frankfurt) | eu-frankfurt |
| East US (Virginia) | na-ashburn |
| West US (Silicon Valley) | na-siliconvalley |

Signature v3

Last updated : 2025-03-10 10:28:36

TencentCloud API authenticates every single request, i.e., the request must be signed using the security credentials in the designated steps. Each request has to contain the signature information (Signature) in the common request parameters and be sent in the specified way and format.

Applying for Security Credentials

The security credential used in this document is a key, which includes a SecretId and a SecretKey. Each user can have up to two pairs of keys.

- SecretId: Used to identify the API caller, which is just like a username.
- SecretKey: Used to authenticate the API caller, which is just like a password.
- **You must keep your security credentials private and avoid disclosure; otherwise, your assets may be compromised. If they are disclosed, please disable them as soon as possible.**

You can apply for the security credentials through the following steps:

1. Log in to the [Tencent Cloud Console](#).
2. Go to the [TencentCloud API Key](#) console page.
3. On the [TencentCloud API Key](#) page, click **Create** to create a SecretId/SecretKey pair.

Using the Resources for Developers

TencentCloud API comes with SDKs for seven commonly used programming languages, including [Python](#), [Java](#), [PHP](#), [Go](#), [NodeJS](#) and [.NET](#). In addition, it provides [API Explorer](#) which enables online call, signature verification, and SDK code generation. If you have any troubles calculating a signature, consult these resources.

TC3-HMAC-SHA256 Signature Algorithm

Compatible with the previous HmacSHA1 and HmacSHA256 signature algorithms, the TC3-HMAC-SHA256 signature algorithm is more secure and supports larger requests and JSON format with better performance. We recommend using TC3-HMAC-SHA256 to calculate the signature.

TencentCloud API supports both GET and POST requests. For the GET method, only the Content-Type: application/x-www-form-urlencoded protocol format is supported. For the POST method, two protocol formats,

Content-Type: application/json and Content-Type: multipart/form-data, are supported. The JSON format is supported by default for all business APIs, and the multipart format is supported only for specific business APIs. In this case, the API cannot be called in JSON format. See the specific business API documentation for more information. The POST method is recommended, as there is no difference in the results of both the methods, but the GET method only supports request packets up to 32 KB.

The following uses querying the list of CVM instances in the Guangzhou region as an example to describe the steps of signature splicing. We chose this API because:

1. CVM is activated by default, and this API is often used;
2. It is read-only and does not change the status of existing resources;
3. It covers many types of parameters, which allows it to be used to demonstrate how to use arrays containing data structures.

In the example, we try to choose common parameters and API parameters that are prone to mistakes. When you actually call an API, please use parameters based on the actual conditions. The parameters vary by API. Do not copy the parameters and values in this example.

Assuming that your SecretId and SecretKey are `AKID*****` and `*****`, respectively, if you want to view the status of the instance in the Guangzhou region whose CVM instance name is "unnamed" and have only one data entry returned, then the request may be:

```
curl -X POST https://cvm.tencentcloudapi.com \
-H "Authorization: TC3-HMAC-SHA256 Credential=AKID*****
*/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host, Signature=a7b85514
48762bd123d6f79e81815e31a92013640a6cef36a08ad4b292a4d2f2" \
-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": ["unnamed"], "Name": "instance-name"}]}'
```

The signature calculation process is explained in detail below.

1. Concatenating the CanonicalRequest String

Concatenate the canonical request string (CanonicalRequest) in the following pseudocode format:

```
CanonicalRequest =
HTTPRequestMethod + '\n' +
CanonicalURI + '\n' +
```



```
CanonicalQueryString + '\n' +
CanonicalHeaders + '\n' +
SignedHeaders + '\n' +
HashedRequestPayload
```

| Field Name | Explanation |
|----------------------|---|
| HTTPRequestMethod | HTTP request method (GET or POST). This example uses <code>POST</code> . |
| CanonicalURI | URI parameter. Slash ("/") is used for API 3.0. |
| CanonicalQueryString | <p>The query string in the URL of the originating HTTP request. This is always an empty string "" for POST requests, and is the string after the question mark (?) for GET requests. For example: <code>Limit=10&Offset=0</code>.</p> <p>Note: <code>CanonicalQueryString</code> must be URL-encoded, referencing RFC3986, the UTF8 character set. We recommend using the programming language library. All special characters must be encoded and capitalized.</p> |
| CanonicalHeaders | <p>Header information for signature calculation, including at least two headers of <code>host</code> and <code>content-type</code>. Custom headers can be added to participate in the signature process to improve the uniqueness and security of the request.</p> <p>Concatenation rules:</p> <ol style="list-style-type: none"> Both the key and value of the header should be converted to lowercase with the leading and trailing spaces removed, so they are concatenated in the format of <code>key:value\n</code> format; If there are multiple headers, they should be sorted in ASCII ascending order by the header keys (lowercase). <p>The calculation result in this example is <code>content-type:application/json; charset=utf-8\nhost:cvm.tencentcloudapi.com\n</code>.</p> <p>Note: <code>content-type</code> must match the actually sent content. In some programming languages, a charset value would be added even if it is not specified. In this case, the request sent is different from the one signed, and the server will return an error indicating signature verification failed.</p> |
| SignedHeaders | <p>Header information for signature calculation, indicating which headers of the request participate in the signature process (they must each individually correspond to the headers in CanonicalHeaders). <code>Content-type</code> and <code>host</code> are required headers.</p> <p>Concatenation rules:</p> <ol style="list-style-type: none"> Both the key and value of the header should be converted to lowercase; If there are multiple headers, they should be sorted in ASCII ascending order by the header keys (lowercase) and separated by semicolons (;). <p>The value in this example is <code>content-type;host</code></p> |
| HashedRequestPayload | Hash value of the request payload (i.e., the body, such as <code>{"Limit": 1, "Filter</code> |

```
[{"Values": ["unnamed"], "Name": "instance-name"}]} in this example
The pseudocode for calculation is
Lowercase(HexEncode(Hash.SHA256(RequestPayload))) by SHA256 hashing the pay
of the HTTP request, performing hexadecimal encoding, and finally converting the encc
string to lowercase letters. For GET requests, RequestPayload is always an empt
string. The calculation result in this example is
99d58dfbc6745f6747f36bfca17dee5e6881dc0428a0a36f96199342bc5b4907
```

According to the rules above, the CanonicalRequest string obtained in the example is as follows:

```
POST
/

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

content-type;host
99d58dfbc6745f6747f36bfca17dee5e6881dc0428a0a36f96199342bc5b4907
```

2. Concatenating the String to Be Signed

The string to sign is concatenated as follows:

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

| Field Name | Explanation |
|------------------|--|
| Algorithm | Signature algorithm, which is currently always TC3-HMAC-SHA256 . |
| RequestTimestamp | Request timestamp, i.e., the value of the common parameter X-TC-Timestamp in request header, which is the UNIX timestamp of the current time in seconds, such as 1551113065 in this example. |
| CredentialScope | Scope of the credential in the format of Date/service/tc3_request , including date, requested service and termination string (tc3_request). Date is a date in UTC time, whose value should match the UTC date converted by the common parameter X-TC-Timestamp ; service is the product name, which should match the domain name of the product called. The calculation result in this example is 2018-10-25/cvm/tc3_request . |

| | |
|------------------------|--|
| HashedCanonicalRequest | <p>Hash value of the CanonicalRequest string concatenated in the steps above. The pseudocode for calculation is Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))</p> <p>The calculation result in this example is</p> <pre>2815843035062fffd6f2a44ea8a34818b0dc46f024b8b3786976a3ad</pre> |
|------------------------|--|

Note:

1. Date has to be calculated from the timestamp "X-TC-Timestamp" and the time zone is UTC+0. If you add the system's local time zone information (such as UTC+8), calls can succeed both day and night but will definitely fail at 00:00. For example, if the timestamp is 1551113065 and the time in UTC+8 is 2019-02-26 00:44:25, the UTC+0 date in the calculated Date value should be 2019-02-25 instead of 2019-02-26.
2. Timestamp must be the same as your current system time, and your system time and standard time must be synced; if the difference between Timestamp and your current system time is larger than five minutes, the request will fail. If your system time is out of sync with the standard time for a while, the request will fail and return a signature expiration error.

According to the preceding rules, the string to be signed obtained in the example is as follows:

```
TC3-HMAC-SHA256
1551113065
2019-02-25/cvm/tc3_request
2815843035062fffd6f2a44ea8a34818b0dc46f024b8b3786976a3adda7a
```

3. Calculating the Signature

1. Calculate the derived signature key with the following pseudocode:

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

| Field Name | Explanation |
|------------|---|
| SecretKey | The original SecretKey, i.e., *****. |
| Date | The Date field information in Credential, such as 2019-02-25 in this example. |

| | |
|---------|---|
| Service | Value in the Service field in <code>Credential</code> , such as <code>cvm</code> in this example. |
|---------|---|

2. Calculate the signature with the following pseudocode:

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

4. Concatenating the Authorization

The Authorization is concatenated as follows:

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

| Field Name | Explanation |
|-----------------|--|
| Algorithm | Signature algorithm, which is always <code>TC3-HMAC-SHA256</code> . |
| SecretId | The SecretId in the key pair, i.e., <code>AKID*****</code> . |
| CredentialScope | Credential scope (see above). The calculation result in this example is <code>2019-02-25/cvm/tc3_request</code> . |
| SignedHeaders | Header information for signature calculation (see above), such as <code>content-type;host</code> in this example. |
| Signature | Signature value. The calculation result in this example is <code>a7b8551448762bd123d6f79e81815e31a92013640a6cef36a08ad4b292a4d2f2</code> . |

According to the rules above, the value obtained in the example is:

```
TC3-HMAC-SHA256 Credential=AKID*****/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host, Signature=a7b8551448762bd123d6f79e81815e31a92013640a6cef36a08ad4b292a4d2f2
```

The following example shows a finished authorization header:

```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKID*****/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host, Signature=a7b8551448762bd123d6f79e81815e31a92013640a6cef36a08ad4b292a4d2f2
```

```
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": ["unnamed"], "Name": "instance-name"}]}
```

5. Signature Demo

When calling API 3.0, you are recommended to use the corresponding Tencent Cloud SDK 3.0 which encapsulates the signature process, enabling you to focus on only the specific APIs provided by the product when developing. See [SDK Center](#) for more information. Currently, the following programming languages are supported:

- [Python](#)
- [Java](#)
- [PHP](#)
- [Go](#)
- [NodeJS](#)
- [.NET](#)

To further explain the signing process, we will use a programming language to implement the process described above. The request domain name, API and parameter values in the sample are used here. This goal of this example is only to provide additional clarification for the signature process, please see the SDK for actual usage.

The final output URL might be: `https://cvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKID*****&Signature=ElIP9YW3pW28FpsEdkXt%2F%2BWcGeI%3D&Timestamp=1465185768&Version=2017-03-12.`

Note: The key in the example is fictitious, and the timestamp is not the current time of the system, so if this URL is opened in the browser or called using commands such as curl, an authentication error will be returned: Signature expired. In order to get a URL that can work properly, you need to replace the SecretId and SecretKey in the example with your real credentials and use the current time of the system as the Timestamp.

Note: In the example below, even if you use the same programming language, the order of the parameters in the URL may be different for each execution. However, the order does not matter, as long as all the parameters are included in the URL and the signature is calculated correctly.

Note: The following code is only applicable to API 3.0. It cannot be directly used in other signature processes. Even with an older API, signature calculation errors may occur due to the differences in details. Please refer to the corresponding documentation.

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;
    private final static String SECRET_ID = "AKID*****";
    private final static String SECRET_KEY = "*****";
    private final static String CT_JSON = "application/json; charset=utf-8";

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

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

    public static void main(String[] args) throws Exception {
        String service = "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");
        // Pay attention to the time zone; otherwise, errors may occur
        sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
        String date = sdf.format(new Date(Long.valueOf(timestamp + "000")));

        // ***** Step 1: Concatenate the CanonicalRequest string *****
    }
}
```

```
String httpRequestMethod = "POST";
String canonicalUri = "/";
String canonicalQueryString = "";
String canonicalHeaders = "content-type:application/json; charset=utf-8\n" + "host:" + host + "\n";
String signedHeaders = "content-type;host";

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

// ***** Step 2: Concatenate the string to sign *****
String credentialScope = date + "/" + service + "/" + "tc3_request";
String hashedCanonicalRequest = sha256Hex(canonicalRequest);
String stringToSign = algorithm + "\n" + timestamp + "\n" + credentialScope + "\n" + hashedCanonicalRequest;
System.out.println(stringToSign);

// ***** Step 3: Calculate the signature *****
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);

// ***** Step 4: Concatenate the 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

# Key Parameters
secret_id = "AKID*****"
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": ["unnamed"], "Name": "instance-name"}]}

# ***** Step 1: Concatenate the CanonicalRequest string *****
http_request_method = "POST"
canonical_uri = "/"
canonical_querystring = ""
ct = "application/json; charset=utf-8"
payload = json.dumps(params)
canonical_headers = "content-type:%s\nhost:%s\n" % (ct, host)
signed_headers = "content-type;host"
hashed_request_payload = hashlib.sha256(payload.encode("utf-8")).hexdigest()
canonical_request = (http_request_method + "\n" +
canonical_uri + "\n" +

```



```
canonical_querystring + "\n" +
canonical_headers + "\n" +
signed_headers + "\n" +
hashed_request_payload)
print(canonical_request)

# ***** Step 2: Concatenate the string to sign *****
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)

# ***** Step 3: Calculate the Signature *****
# Function for computing signature digest
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)

# ***** Step 4: Concatenate the 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"
    "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() {
    secretId := "AKID*****"
    secretKey := "*****"
    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 := "content-type:application/json; charset=utf-8\n" + "host:" +
        host + "\n"
    signedHeaders := "content-type;host"
    payload := `{"Limit": 1, "Filters": [{"Values": ["unnamed"], "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
$secretId = "AKID*****";
$secretKey = "*****";
$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 = "content-type:application/json; charset=utf-8\n"."host:". $host. "\n";
$signedHeaders = "content-type;host";
$payload = '{"Limit": 1, "Filters": [{"Values": ["unnamed"], "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."/".$credentialScope
.", SignedHeaders=content-type;host, Signature=".$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'

# Key Parameters
secret_id = 'AKID*****'
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')

# ***** Step 1: Concatenate the CanonicalRequest string *****
http_request_method = 'POST'
canonical_uri = '/'
canonical_querystring = ''
```

```
canonical_headers = "content-type:application/json; charset=utf-8\nhost:#{host}\n"\nsigned_headers = 'content-type;host'\n# params = { 'Limit' => 1, 'Filters' => [{ 'Name' => 'instance-name', 'Values' => ['unnamed'] }] }\n# payload = JSON.generate(params, { 'ascii_only' => true, 'space' => ' ' })\n# json will generate in random order, to get specified result in example, we hard-code it here.\npayload = '{"Limit": 1, "Filters": [{"Values": ["unnamed"], "Name": "instance-name"}]}'\nhashed_request_payload = Digest::SHA256.hexdigest(payload)\ncanonical_request = [\n  http_request_method,\n  canonical_uri,\n  canonical_querystring,\n  canonical_headers,\n  signed_headers,\n  hashed_request_payload,\n].join("\n")\n\nputs canonical_request\n\n# ***** Step 2: Concatenate the string to sign *****\ncredential_scope = date + '/' + service + '/' + 'tc3_request'\nhashed_request_payload = Digest::SHA256.hexdigest(canonical_request)\nstring_to_sign = [\n  algorithm,\n  timestamp.to_s,\n  credential_scope,\n  hashed_request_payload,\n].join("\n")\nputs string_to_sign\n\n# ***** Step 3: Calculate the Signature *****\ndigest = OpenSSL::Digest.new('sha256')\nsecret_date = OpenSSL::HMAC.digest(digest, 'TC3' + secret_key, date)\nsecret_service = OpenSSL::HMAC.digest(digest, secret_date, service)\nsecret_signing = OpenSSL::HMAC.digest(digest, secret_service, 'tc3_request')\nsignature = OpenSSL::HMAC.hexdigest(digest, secret_signing, string_to_sign)\nputs signature\n\n# ***** Step 4: Concatenate the Authorization *****\nauthorization = "#{algorithm} Credential=#{secret_id}/#{credential_scope}, Signed Headers=#{signed_headers}, Signature=#{signature}"\nputs authorization\n\nputs '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, Mi
```

```
dpointRounding.AwayFromZero) / 1000;
// ***** Step 1: Concatenate the CanonicalRequest string *****
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";
string signedHeaders = "content-type;host";
string hashedRequestPayload = SHA256Hex(requestPayload);
string canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload;
Console.WriteLine(canonicalRequest);
Console.WriteLine("-----");

// ***** Step 2: Concatenate the string to sign *****
string credentialScope = datestr + "/" + service + "/" + "tc3_request";
string hashedCanonicalRequest = SHA256Hex(canonicalRequest);
string stringToSign = algorithm + "\n" + requestTimestamp.ToString() + "\n" + cre
dentialScope + "\n" + hashedCanonicalRequest;
Console.WriteLine(stringToSign);
Console.WriteLine("-----");

// ***** Step 3: Calculate the signature *****
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_requ
est"));
byte[] signatureBytes = HmacSHA256(secretSigning, Encoding.UTF8.GetBytes(stringTo
Sign));
string signature = BitConverter.ToString(signatureBytes).Replace("-", "").ToLower
();
Console.WriteLine(signature);
Console.WriteLine("-----");

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



```
Console.WriteLine("-----");

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)
{
    // SecretID and SecretKey
    string SECRET_ID = "AKID*****";
    string SECRET_KEY = "*****";

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

    // The timestamp `2019-02-26 00:44:25` used here is only for reference. In a project, use the following parameter:
    // DateTime date = DateTime.UtcNow;
    // Enter the correct time zone. We recommend using UTC timestamp to avoid errors.
    DateTime date = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc).AddSeconds(1551113065);
    string requestPayload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"unnamed\"], \"Name\": \"instance-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(){

const SECRET_ID = "AKID*****"
const 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
const date = getDate(timestamp)

// ***** Step 1: Concatenate the CanonicalRequest string *****
const signedHeaders = "content-type;host"

const payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"unnamed\"], \"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"

const canonicalRequest = httpRequestMethod + "\n"
```

```
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload
console.log(canonicalRequest)
console.log("-----")

// ***** Step 2: Concatenate the string to sign *****
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)
console.log("-----")

// ***** Step 3: Calculate the signature *****
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)
console.log("-----")

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

const Call_Information = '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(Call_Information)
}
main()
```

C++

```
#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++)
    {
        sprintf(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 OPENSSSL_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 OPENSSSL_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()
{
string SECRET_ID = "AKID*****";
string 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);

// ***** Step 1: Concatenate the CanonicalRequest string *****
string httpRequestMethod = "POST";
string canonicalUri = "/";
string canonicalQueryString = "";
string canonicalHeaders = "content-type:application/json; charset=utf-8\nhost:" +
host + "\n";
string signedHeaders = "content-type;host";
string payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"unnamed\"], \"Name\": \"instance-name\"}] }";
string hashedRequestPayload = sha256Hex(payload);
string canonicalRequest = httpRequestMethod + "\n" + canonicalUri + "\n" + canonicalQueryString + "\n"
+ canonicalHeaders + "\n" + signedHeaders + "\n" + hashedRequestPayload;
cout << canonicalRequest << endl;
cout << "-----" << endl;

// ***** Step 2: Concatenate the string to sign *****
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" + hashedCanonicalRequest;
cout << stringToSign << endl;
cout << "-----" << endl;

// ***** Step 3: Calculate the signature *****
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;
cout << "-----" << endl;

// ***** Step 4: Concatenate the Authorization *****
string authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + creden
tialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
cout << authorization << endl;
cout << "-----" << endl;

string headers = "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 << headers << endl;
return 0;
};

```

Signature Failure

The following situational error codes for signature failure may occur. Please resolve the errors accordingly.

| Error Code | Description |
|------------------------------|---|
| AuthFailure.SignatureExpire | Signature expired. Timestamp and server time cannot differ by more than five minutes. |
| AuthFailure.SecretIdNotFound | The key does not exist. Please go to the console to check whether it is disabled or you copied fewer or more characters. |
| AuthFailure.SignatureFailure | Signature error. It is possible that the signature was calculated incorrectly, the signature does not match the content actually sent, or the SecretKey is incorrect. |
| AuthFailure.TokenFailure | Temporary certificate token error. |
| AuthFailure.InvalidSecretId | Invalid key (not a TencentCloud API key type). |

Signature

Last updated : 2025-03-10 10:28:37

Tencent Cloud API authenticates each access request, i.e. each request needs to include authentication information (Signature) in the common parameters to verify the identity of the requester.

The Signature is generated by the security credentials which include SecretId and SecretKey. If you don't have the security credentials yet, go to the [TencentCloud API Key](#) page to apply for them; otherwise, you cannot invoke the TencentCloud API.

1. Applying for Security Credentials

Before using the TencentCloud API for the first time, go to the [TencentCloud API Key](#) page to apply for security credentials.

Security credentials consist of SecretId and SecretKey:

- SecretId is used to identify the API requester.
- SecretKey is used to encrypt the signature string and verify it on the server.
- **You must keep your security credentials private and avoid disclosure.**

You can apply for the security credentials through the following steps:

1. Log in to the [Tencent Cloud Console](#).
2. Go to the [TencentCloud API Key](#) page.
3. On the [API Key Management](#) page, click **Create Key** to create a SecretId/SecretKey pair.

Note: Each account can have up to two pairs of SecretId/SecretKey.

2. Generating a Signature

With the SecretId and SecretKey, a signature can be generated. The following describes how to generate a signature:

Assume that the SecretId and SecretKey are:

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

Note: This is just an example. For actual operations, please use your own SecretId and SecretKey.

Take the Cloud Virtual Machine's request to view the instance list (DescribeInstances) as an example. When you invoke this API, the request parameters may be as follows:

| Parameter name | Description | Parameter value |
|----------------|--------------------------------------|-------------------|
| Action | Method name | DescribeInstances |
| SecretId | Key ID | AKID***** |
| Timestamp | Current timestamp | 1465185768 |
| Nonce | Random positive integer | 11886 |
| Region | Region where the instance is located | ap-guangzhou |
| InstanceIds.0 | ID of the instance to query | ins-09dx96dg |
| Offset | Offset | 0 |
| Limit | Allowed maximum output | 20 |
| Version | API version number | 2017-03-12 |

2.1. Sorting Parameters

First, sort all the request parameters in an ascending lexicographical order (ASCII code) by their names. Notes: (1) Parameters are sorted by their names instead of their values; (2) The parameters are sorted based on ASCII code, not in an alphabetical order or by values. For example, InstanceIds.2 should be arranged after InstanceIds.12. You can complete the sorting process using a sorting function in a programming language, such as the ksort function in PHP. The parameters in the example are sorted as follows:

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

When developing in another programming language, you can sort these sample parameters and it will work as long as you obtain the same results.

2.2. Concatenating a Request String

This step generates a request string.

Format the request parameters sorted in the previous step into the form of "parameter name"="parameter value". For example, for the Action parameter, its parameter name is "Action" and its parameter value is "DescribeInstances", so it will become Action=DescribeInstances after formatted.

Note: The "parameter value" is the original value but not the value after URL encoding.

Then, concatenate the formatted parameters with "&". The resulting request string is as follows:

```
Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0
&Region=ap-guangzhou&SecretId=AKID*****&Timestamp=1465
185768&Version=2017-03-12
```

2.3. Concatenating the Signature Original String

This step generates a signature original string.

The signature original string consists of the following parameters:

1. HTTP method: POST and GET modes are supported, and GET is used here for the request. Please note that the method name should be in all capital letters.
2. Request server: the domain name of the request to view the list of instances (DescribeInstances) is cvm.tencentcloudapi.com. The actual request domain name varies by the module to which the API belongs. For more information, see the instructions of the specific API.
3. Request path: The request path in the current version of TencentCloud API is fixed to /.
4. Request string: the request string generated in the previous step.

The concatenation rule of the signature original string is: Request method + request host + request path + ? + request string

The concatenation result of the example is:

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

2.4. Generating a Signature String

This step generates a signature string.

First, use the HMAC-SHA1 algorithm to sign the **signature original string** obtained in the previous step, and then

encode the generated signature using Base64 to obtain the final signature.

The specific code is as follows with the PHP language being used as an example:

```
$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;
```

The final signature is:

```
7RAM2xfNMO9EiVTNmPg06MRnCvQ=
```

When developing in another programming language, you can sign and verify the original in the example above and it works as long as you get the same results.

3. Encoding a Signature String

The generated signature string cannot be directly used as a request parameter and must be URL encoded.

For example, if the signature string generated in the previous step is 7RAM2xfNMO9EiVTNmPg06MRnCvQ=, the final signature string request parameter (Signature) is 7RAM2xfNMO9EiVTNmPg06MRnCvQ%3D, which will be used to generate the final request URL.

Note: If your request method is GET, or the request method is POST and the Content-Type is application/x-www-form-urlencoded, then all the request parameter values need to be URL encoded (except the parameter key and the symbol of =) when sending the request. Non-ASCII characters need to be encoded with UTF-8 before URL encoding.

Note: The network libraries of some programming languages automatically URL encode all parameters, in which case there is no need to URL encode the signature string; otherwise, two rounds of URL encoding will cause the signature to fail.

Note: Other parameter values also need to be encoded using [RFC 3986](#). Use %XY in percent-encoding for special characters such as Chinese characters, where "X" and "Y" are hexadecimal characters (0-9 and uppercase A-F), and using lowercase will cause an error.

4. Signature Failure

The following situational error codes for signature failure may occur. Please resolve the errors accordingly.

| Error code | Error description |
|------------------------------|---|
| AuthFailure.SignatureExpire | The signature is expired |
| AuthFailure.SecretIdNotFound | The key does not exist |
| AuthFailure.SignatureFailure | Signature error |
| AuthFailure.TokenFailure | Token error |
| AuthFailure.InvalidSecretId | Invalid key (not a TencentCloud API key type) |

5. Signature Demo

When calling API 3.0, you are recommended to use the corresponding Tencent Cloud SDK 3.0 which encapsulates the signature process, enabling you to focus on only the specific APIs provided by the product when developing. See [SDK Center](#) for more information. Currently, the following programming languages are supported:

- [Python](#)
- [Java](#)
- [PHP](#)
- [Go](#)
- [NodeJS](#)
- [.NET](#)

To further explain the signing process, we will use a programming language to implement the process described above. The request domain name, API and parameter values in the sample are used here. This goal of this example is only to provide additional clarification for the signature process, please see the SDK for actual usage.

The final output URL might be: `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` .

Note: The key in the example is fictitious, and the timestamp is not the current time of the system, so if this URL is opened in the browser or called using commands such as curl, an authentication error will be returned: Signature expired. In order to get a URL that can work properly, you need to replace the SecretId and SecretKey in the example with your real credentials and use the current time of the system as the Timestamp.

Note: In the example below, even if you use the same programming language, the order of the parameters in the URL may be different for each execution. However, the order does not matter, as long as all the parameters are included in the URL and the signature is calculated correctly.

Note: The following code is only applicable to API 3.0. It cannot be directly used in other signature processes. Even with an older API, signature calculation errors may occur due to the differences in details. Please refer to the corresponding documentation.

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/?");
        // When signing, the parameters need to be sorted in lexicographical order. TreeMap
        // is used here to guarantee the correct order.
        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/?");
        // There is no requirement for the order of the parameters in the actual request
        // URL.
        for (String k : params.keySet()) {
```

```
// The request string needs to be URL encoded. As the Key is all in English letters, only the value is URL encoded here.
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 enable s automatic sorting
    // A random number should be used when actually calling, for example: params.put ("Nonce", new Random().nextInt (java.lang.Integer.MAX_VALUE));
    params.put ("Nonce", 11886); // Common parameter
    // The current time of the system should be used when actually calling, for example: params.put ("Timestamp", System.currentTimeMillis() / 1000);
    params.put ("Timestamp", 1465185768); // Common parameter
    params.put ("SecretId", "AKID*****"); // Common parameter
    params.put ("Action", "DescribeInstances"); // Common parameter
    params.put ("Version", "2017-03-12"); // Common parameter
    params.put ("Region", "ap-guangzhou"); // Common parameter
    params.put ("Limit", 20); // Business parameter
    params.put ("Offset", 0); // Business parameter
    params.put ("InstanceIds.0", "ins-09dx96dg"); // Business parameter
    params.put ("Signature", sign (getStringToSign (params), "*****", "HmacSHA1")); // Common parameter
    System.out.println (getUrl (params));
}
}
```

Python

Note: If running in a Python 2 environment, the following requests dependency package must be installed first: `pip`

`install requests`.

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

import requests

secret_id = "AKID*****"
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"])
    # An actual invocation would occur here, which may incur fees after success
    # resp = requests.get("https://" + endpoint, params=data)
    # print(resp.url)
```

Golang

```
package main

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

func main() {
    secretId := "AKID*****"
    secretKey := "*****"
```

```
params := map[string]string{
    "Nonce": "11886",
    "Timestamp": "1465185768",
    "Region": "ap-guangzhou",
    "SecretId": secretId,
    "Version": "2017-03-12",
    "Action": "DescribeInstances",
    "InstanceIds.0": "ins-09dx96dg",
    "Limit": "20",
    "Offset": "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
$secretId = "AKID*****";
$secretKey = "*****";
$params["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'

secret_id = "AKID*****"
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, string 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)
{
string SECRET_ID = "AKID*****";
string SECRET_KEY = "*****";

string endpoint = "cvm.tencentcloudapi.com";
string region = "ap-guangzhou";
string action = "DescribeInstances";
string version = "2017-03-12";
double RequestTimestamp = 1465185768;
// 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>(p
aram, StringComparer.Ordinal);
string sigInParam = MakeSignPlainText(headers, "GET", endpoint, "/");
Console.WriteLine(sigInParam);
string sigOutParam = Sign(SECRET_KEY, sigInParam);
```

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

string result = "https://cvm.tencentcloudapi.com/?";
foreach (KeyValuePair<string, string> kv in headers)
{
    result += WebUtility.UrlEncode(kv.Key) + "=" + WebUtility.UrlEncode(kv.Value) +
"&";
}
result += WebUtility.UrlEncode("Signature") + "=" + WebUtility.UrlEncode(sigOutPa
ram);
Console.WriteLine("GET " + result);
}
}
```

NodeJS

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

function get_req_url(params, endpoint){
    params['Signature'] = escape(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(){
const SECRET_ID = "AKID*****"
const SECRET_KEY = "*****"

const endpoint = "cvm.tencentcloudapi.com"
const Region = "ap-guangzhou"
const Version = "2017-03-12"
const Action = "DescribeInstances"
const Timestamp = 1465185768
// const Timestamp = Math.round(Date.now() / 1000)
const Nonce = 11886
//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;

strParam = sort_params(params)

const reqMethod = "GET";
const path = "/";
strSign = formatSignString(reqMethod, endpoint, path, strParam)
console.log(strSign)
console.log("-----")

params['Signature'] = sha1(SECRET_KEY, strSign)
console.log(params['Signature'])
console.log("-----")

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


Responses

Last updated : 2024-11-25 17:03:37

Response for Successful Requests

For example, when calling CAM API (version: 2017-03-12) to view the status of instances (DescribeInstancesStatus), if the request has succeeded, you may see the response as shown below:

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

- The API will return `Response` , which contains `RequestId` , as long as it processes the request. It does not matter if the request is successful or not.
- `RequestId` is the unique ID of an API request. Contact us with this ID when an exception occurs.
- Except for the fixed fields, all fields are action-specified. For the definitions of action-specified fields, see the corresponding API documentation. In this example, `TotalCount` and `InstanceStatusSet` are the fields specified by the API `DescribeInstancesStatus` . `0` `TotalCount` means that the requester owns 0 CVM instance so the `InstanceStatusSet` is empty.

Response for Failed Requests

If the request has failed, you may see the response as shown below:

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

- The presence of the `Error` field indicates that the request has failed. A response for a failed request will include `Error`, `Code` and `Message` fields.
- `Code` is the code of the error that helps you identify the cause and solution. There are two types of error codes so you may find the code in either common error codes or API-specified error codes.
- `Message` explains the cause of the error. Note that the returned messages are subject to service updates. The information the messages provide may not be up-to-date and should not be the only source of reference.
- `RequestId` is the unique ID of an API request. Contact us with this ID when an exception occurs.

Common Error Codes

If there is an `Error` field in the response, it means that the API call failed. The `Code` field in `Error` indicates the error code. The following table lists the common error codes that all actions can return.

| Error Code | Description |
|--|---|
| <code>AuthFailure.InvalidSecretId</code> | Invalid key (not a TencentCloud API key type). |
| <code>AuthFailure.MFAFailure</code> | MFA failed. |
| <code>AuthFailure.SecretIdNotFound</code> | The key does not exist. |
| <code>AuthFailure.SignatureExpire</code> | Signature expired. |
| <code>AuthFailure.SignatureFailure</code> | Signature error. |
| <code>AuthFailure.TokenFailure</code> | Token error. |
| <code>AuthFailure.UnauthorizedOperation</code> | The request does not have CAM authorization. |
| <code>DryRunOperation</code> | DryRun Operation. It means that the request would have succeeded, but the <code>DryRun</code> parameter was used. |
| <code>FailedOperation</code> | Operation failed. |
| <code>InternalError</code> | Internal error. |
| <code>InvalidAction</code> | The API does not exist. |
| <code>InvalidParameter</code> | Incorrect parameter. |
| <code>InvalidParameterValue</code> | Invalid parameter value. |
| <code>LimitExceeded</code> | Quota limit exceeded. |
| <code>MissingParameter</code> | A parameter is missing. |

| | |
|-----------------------|---|
| NoSuchVersion | The API version does not exist. |
| RequestLimitExceeded | The number of requests exceeds the frequency limit. |
| ResourceInUse | Resource is in use. |
| ResourceInsufficient | Insufficient resource. |
| ResourceNotFound | The resource does not exist. |
| ResourceUnavailable | Resource is unavailable. |
| UnauthorizedOperation | Unauthorized operation. |
| UnknownParameter | Unknown parameter. |
| UnsupportedOperation | Unsupported operation. |
| UnsupportedProtocol | HTTPS request method error. Only GET and POST requests are supported. |
| UnsupportedRegion | API does not support the requested region. |

Image Moderation APIs

ImageModeration

Last updated : 2025-03-11 22:47:39

1. API Description

Domain name for API request: ims.intl.tencentcloudapi.com.

This API is used to submit an image for smart moderation. Before using it, you need to log in to the console with the Tencent Cloud root account [to activate IMS](#) and adjust the business configuration.

API use instructions

- Go to the "[CMS console - IMS](#)" to activate IMS.
- This API is a paid API. For its billing mode, see [IMS Pricing](#).

API feature description

- It can detect images or links and recognize content that may be offensive, unsafe, or inappropriate based on the deep learning technology;
- It can capture frames from or split GIF/long images for detection;
- It can recognize various non-compliant scenarios, including vulgarity, law or regulation violations, pornography, and advertising;
- It can detect multiple types of objects (such as object, advertising logo, and QR code) and recognize text in images based on OCR;
- It allows you to customize moderation policies based on different business scenarios;
- You can select image risk libraries to filter non-compliant images of custom types (currently, only blacklist configuration is supported);
- You can associate account or device information when moderating an image to recognize the account or device involved.

API call description

- Supported image file size: **< 5 MB**
- Supported image file resolution: **a resolution of 256x256 or higher** is recommended; otherwise, the recognition effect may be affected.
- Supported image file formats: PNG, JPG, JPEG, BMP, GIF, and WEBP.
- Supported image URL transfer protocols: HTTP and HTTPS.

- If you pass in the access URL of an image, you need to **limit the image download time to 3 seconds**. To ensure the stability and reliability of the image to be detected, we recommend you use Tencent Cloud COS for storage or CDN for caching.
- Default API request rate limit: **100 requests/sec**. If this limit is exceeded, an error will be reported.

A maximum of 100 requests can be initiated per second for this API.

We recommend you to use API Explorer

[Try it](#)

API Explorer provides a range of capabilities, including online call, signature authentication, SDK code generation, and API quick search. It enables you to view the request, response, and auto-generated examples.

2. Input Parameters

The following request parameter list only provides API request parameters and some common parameters. For the complete common parameter list, see [Common Request Parameters](#).

| Parameter Name | Required | Type | Description |
|----------------|----------|--------|---|
| Action | Yes | String | Common Params . The value used for this API: ImageModeration. |
| Version | Yes | String | Common Params . The value used for this API: 2020-12-29. |
| Region | Yes | String | Common Params . For more information, please see the list of regions supported by the product. This API only supports: ap-mumbai, ap-singapore, eu-frankfurt. |
| BizType | No | String | This field indicates the specific number of the policy, which is used for API scheduling and can be configured in the CMS console. If the <code>Biztype</code> parameter is passed in, a moderation policy will be used based on the business scenario; otherwise, the default moderation policy will be used. Note: <code>Biztype</code> can contain 3-32 digits, letters, and underscores; different <code>Biztype</code> values are associated with different business scenarios and moderation policies, so you need to verify the <code>Biztype</code> before calling this API. |
| DataId | No | String | This field indicates the data ID assigned by you to the object to be detected for easier file identification and management. It can contain up to 64 letters, digits, and special symbols (<code>_ - @#</code>). |
| | | | |

| | | | |
|-------------|----|---------|---|
| FileContent | No | String | <p>This field indicates the Base64 encoding of the image to be detected. The image size cannot exceed 5 MB. A resolution of 256x256 or higher is recommended; otherwise, the recognition effect may be affected.</p> <p>Note: you must enter a value for either this field or <code>FileUrl</code> .</p> |
| FileUrl | No | String | <p>URL of the image to moderate. It supports PNG, JPG, JPEG, BMP, GIF AND WEBP files. The file cannot exceed 5 MB and the resolution should not below 256*246. The default timeout period is 3 seconds. Note that redirection URLs may be blocked by security policies. In this case, an error message will return. For example, if an HTTP request gets the 302 code, the error <code>ResourceUnavailable.ImageDownloadError</code> is returned.</p> <p>**Either <code>FileUrl</code> or <code>FileContent</code> must be specified.</p> |
| Interval | No | Integer | <p>For GIF/long image detection only. This field indicates the GIF frame capturing frequency (the image interval for capturing a frame for detection). For long images, you should round the width:height ratio as the total number of images to be split. The default value is 0, where only the first frame of the GIF image will be detected, and the long image will not be split.</p> <p>Note: the <code>Interval</code> and <code>MaxFrames</code> parameters need to be used in combination; for example, if <code>Interval</code> is 3 and <code>MaxFrames</code> is 400 , the GIF/long image will be detected once every two frames for up to 400 frames.</p> |
| MaxFrames | No | Integer | <p>For GIF/long image detection only. This field indicates the maximum number of frames that can be captured. The default value is 1, where only the first frame of the input GIF image will be detected, and the long image will not be split (which may cause a processing timeout).</p> <p>Note: the <code>Interval</code> and <code>MaxFrames</code> parameters need to be used in combination; for example, if <code>Interval</code> is 3 and <code>MaxFrames</code> is 400 , the GIF/long image will be detected once every two frames for up to 400 frames.</p> |
| User | No | User | <p>This field indicates the information of the user that corresponds to the object to be detected. It can be passed in to identify the user involved in the violation.</p> |
| Device | No | Device | <p>This field indicates the information of the device that corresponds to the object to be detected. It can be passed in to identify the device involved in the violation.</p> |

3. Output Parameters

| Parameter Name | Type | Description |
|----------------|---------------------------------------|---|
| Suggestion | String | This field is used to return the operation suggestion for the <code>Label</code> tag. When you get the determination result, the returned value indicates the operation suggested by the system. We recommend you handle different types of violations and suggestions according to your business needs. Returned values: Block , Review , Pass . |
| Label | String | This field is used to return the maliciousness tag with the highest priority in the detection result (<code>LabelResults</code>), which represents the moderation result suggested by the model. We recommend you handle different types of violations and suggestions according to your business needs. Returned values: Normal : normal; Porn : pornographic; Abuse : abusive; Ad : advertising; Custom : custom type of non-compliant content and other offensive, unsafe, or inappropriate types of content. |
| SubLabel | String | This field is used to return the subtag name under the maliciousness tag with the highest priority hit by the detection result, such as <i>Porn-SexBehavior</i> . If no subtag is hit, an empty string will be returned. |
| Score | Integer | Confidence score of the under the current label. Value range: 0 (the lowest confidence) to 100 (the highest confidence). For example, <i>Porn 99</i> indicates that the image is highly likely to be pornographic, while <i>Porn 0</i> indicates that the image is not pornographic. |
| LabelResults | Array of LabelResult | This field is used to return the detailed recognition result for the maliciousness tag hit by the categorization model, such as porn, advertising, or any other offensive, unsafe, or inappropriate type of content. Note: this field may return null, indicating that no valid values can be obtained. |
| ObjectResults | Array of ObjectResult | This field is used to return the detailed detection result of the object detection model, including the tag name hit by the content such as object, advertising logo, or QR code, tag score, coordinate information, scenario recognition result, and operation suggestion. For more information on the returned value, see the description of the <code>ObjectResults</code> data structure. |

| | | |
|--------------------|--|---|
| | | Note: this field may return null, indicating that no valid values can be obtained. |
| OcrResults | Array of OcrResult | This field is used to return the detailed text OCR result, including the text coordinate information, text recognition result, and operation suggestion. For more information on the returned value, see the description of the <code>OcrResults</code> data structure. Note: this field may return null, indicating that no valid values can be obtained. |
| LibResults | Array of LibResult | This field is used to return the result of recognition based on image risk libraries (blocklist and allowlist). For more information on the returned value, see the description of the <code>LibResults</code> data structure. Note: currently, you cannot customize image risk libraries . Note: this field may return null, indicating that no valid values can be obtained. |
| DataId | String | This field is used to return the <code>DataId</code> in the request parameters that correspond to the detected object. |
| BizType | String | This field is used to return the <code>BizType</code> in the request parameters that correspond to the detected object. |
| Extra | String | This field is used to return the additional information (Extra) configured based on your needs. If it is not configured, an empty value will be returned by default. Note: the returned information varies by customer or <code>Biztype</code> . If you need to configure this field, submit a ticket or contact the aftersales service for assistance. Note: this field may return null, indicating that no valid values can be obtained. |
| FileMD5 | String | This field is used to return the MD5 checksum of the detected object for easier verification of the file integrity. |
| RecognitionResults | Array of RecognitionResult | Image recognition result, including the hit tags, confidence and location. Note: This field may return <code>null</code> , indicating that no valid values can be obtained. |
| RequestId | String | The unique request ID, generated by the server, will be returned for every request (if the request fails to reach the server for other reasons, the request will not obtain a RequestId). RequestId is required for locating a problem. |

4. Example

Example1 Detecting Image Synchronization

Input Example

```
POST / HTTP/1.1
Host: ims.intl.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: ImageModeration
<common request parameters>

{
  "BizType": "182600012300002017",
  "DataId": "a61237dd-c2a0-43e7-a3da-d27022d39ba7",
  "FileUrl": "https://cmstest-123.cos.ap-guangzhou.myqcloud.com/image.jpg"
}
```

Output Example

```
{
  "Response": {
    "RequestId": "a61237dd-c2a0-43e7-a3da-d27022d39ba7",
    "DataId": "a61237dd-c2a0-43e7-a3da-d27022d39ba7",
    "BizType": "182600012300002017",
    "Suggestion": "Block",
    "FileMD5": "",
    "Label": "Porn",
    "SubLabel": "SexyBehavior",
    "Score": 90,
    "LabelResults": [
      {
        "Scene": "Porn",
        "Suggestion": "Block",
        "Label": "Porn",
        "SubLabel": "SexyBehavior",
        "Score": 90,
        "Details": []
      }
    ],
    "ObjectResults": [
      {
        "Scene": "QrCode",
        "Suggestion": "Block",
        "Label": "Ad",

```

```
"SubLabel": "",
"Score": 100,
"Names": [
  "QRCODE"
],
"Details": [
  {
    "Id": 0,
    "Name": "QRCODE",
    "Value": "https://test.com/test",
    "SubLabel": "QRCODE",
    "Score": 100,
    "Location": {
      "X": 155.01746,
      "Y": 396.01746,
      "Width": 769.9824,
      "Height": 769.98254,
      "Rotate": 0
    }
  }
]
"OcrResults": [],
"LibResults": [],
"RecognitionResults": [],
"Extra": ""
}
```

5. Developer Resources

SDK

TencentCloud API 3.0 integrates SDKs that support various programming languages to make it easier for you to call APIs.

- [Tencent Cloud SDK 3.0 for Python](#)
- [Tencent Cloud SDK 3.0 for Java](#)
- [Tencent Cloud SDK 3.0 for PHP](#)
- [Tencent Cloud SDK 3.0 for Go](#)
- [Tencent Cloud SDK 3.0 for Node.js](#)
- [Tencent Cloud SDK 3.0 for .NET](#)

- [Tencent Cloud SDK 3.0 for C++](#)

Command Line Interface

- [Tencent Cloud CLI 3.0](#)

6. Error Code

The following only lists the error codes related to the API business logic. For other error codes, see [Common Error Codes](#).

| Error Code | Description |
|--|--|
| DryRunOperation | <code>DryRun</code> Operation. It means that the request would have succeeded, but the <code>DryRun</code> parameter was used. |
| FailedOperation | The operation failed. |
| InternalError | An internal error occurred. |
| InternalError.InternalError | An internal error occurred. |
| InvalidParameter | The parameter is incorrect. |
| InvalidParameter.ImageAspectRatioTooLarge | |
| InvalidParameter.ImageDataTooSmall | |
| InvalidParameter.ImageSizeTooSmall | The image resolution is too low. |
| InvalidParameter.InvalidImageContent | The image content is incorrect. |
| InvalidParameter.InvalidParameter | The parameter is invalid. |
| InvalidParameterValue | The parameter value is incorrect. |
| InvalidParameterValue.EmptyImageContent | The image content parameter is empty. |
| InvalidParameterValue.ImageSizeTooSmall | The image resolution is too low. |
| InvalidParameterValue.InvalidContent | <code>FileContent</code> and <code>FileUrl</code> are empty. |
| InvalidParameterValue.InvalidDataId | The <code>DataId</code> format is incorrect. |
| InvalidParameterValue.InvalidFileContentSize | The image file content size is exceptional. |
| InvalidParameterValue.InvalidImageContent | The image content is incorrect. |

| | |
|---|--|
| InvalidParameterValue.InvalidParameter | The parameter value is incorrect. |
| LimitExceeded | The quota limit is exceeded. |
| MissingParameter | The parameter is missing. |
| OperationDenied | The operation was denied. |
| RequestLimitExceeded | The number of requests exceeds the frequency limit. |
| ResourceInUse | The resource is in use. |
| ResourceInsufficient | The resource is insufficient. |
| ResourceNotFound | The resource does not exist. |
| ResourceUnavailable | The resource is unavailable. |
| ResourceUnavailable.ImageDownloadError | Image file download failed. |
| ResourceUnavailable.InvalidImageContent | The image resource is incorrect. |
| ResourceUnavailable.ModelCallFailed | |
| ResourcesSoldOut | The resources have been sold out. |
| UnauthorizedOperation | The operation is unauthorized. |
| UnauthorizedOperation.Unauthorized | Operation not authorized/No valid package/The account is overdue |
| UnknownParameter | The parameter is unknown. |
| UnsupportedOperation | The operation is not supported. |

Data Types

Last updated : 2024-11-25 17:03:39

Device

Indicates the information of the business user's device

Used by actions: ImageModeration.

| Name | Type | Required | Description |
|----------|---------|----------|--|
| Ip | String | No | This field indicates the IP address of the business user's device and supports recording both IPv4 and IPv6 addresses. It needs to be used together with the <code>IpType</code> parameter. |
| Mac | String | No | This field indicates the MAC address of the business user, which makes it easier to identify and manage devices. Its format and value are the same as those of a standard MAC address. |
| TokenId | String | No | <i>In beta test. Stay tuned.</i> |
| DeviceId | String | No | <i>In beta test. Stay tuned.</i> |
| IMEI | String | No | This field indicates the international mobile equipment identity (IMEI) number of the business user's device, which can be used to identify each mobile communication device such as mobile phone for easier device identification and management. Note: the format is 15–17 digits . |
| IDFA | String | No | For iOS devices: this field indicates the identifier for advertisers (IDFA) of the business user, which is provided by Apple to identify the user and contains a hexadecimal string of 32 digits and letters. Note: as iOS 14 has been updated by Apple to allow users to manually enable or disable IDFA since 2021, the validity of this string may be reduced. |
| IDFV | String | No | For iOS devices: this field indicates the identifier for vendors (IDFV) of the business user, which is provided by Apple to identify the app vendor and contains a hexadecimal string of 32 digits and letters. It can be used to uniquely identify a device. |
| IpType | Integer | No | This field indicates the type of the recorded IP address. Valid values: 0 (IPv4 address), 1 (IPv6 address). It needs to be used together with the |

`IpType` parameter.

LabelDetailItem

Returns the details of the subtag hit by the categorization model

Used by actions: ImageModeration.

| Name | Type | Description |
|-------|---------|---|
| Id | Integer | This field is used to return the ID of the recognized object for easier recognition and distinction. Note: this field may return null, indicating that no valid values can be obtained. |
| Name | String | This field is used to return the hit subtag name. Note: this field may return null, indicating that no valid values can be obtained. |
| Score | Integer | This field is used to return the hit score of the subtag. Value range: 0–100 ; for example, <i>Porn-SexBehavior 99</i> indicates that the hit score of the <i>Porn-SexBehavior</i> tag for the recognized content is 99. Note: this field may return null, indicating that no valid values can be obtained. |

LabelResult

Hit result of the categorization model

Used by actions: ImageModeration.

| Name | Type | Description |
|------------|--------|--|
| Scene | String | This field is used to return the scenario result recognized by the model, such as advertising, pornographic, and harmful. |
| Suggestion | String | This field is used to return the operation suggestion for the current maliciousness tag. When you get the determination result, the returned value indicates the operation suggested by the system. We recommend you handle different types of violations and suggestions according to your business needs. Returned values: Block , Review , Pass . |
| Label | String | This field is used to return the maliciousness tag in the detection result. Returned values: Normal : normal; Porn : pornographic; Abuse : abusive; Ad : |

| | | |
|----------|--|--|
| | | advertising; Custom : custom type of non-compliant content and other offensive, unsafe, or inappropriate types of content. |
| SubLabel | String | This field is used to return the detection result for a subtag under the maliciousness tag, such as <i>Porn-SexBehavior</i> . |
| Score | Integer | Confidence score of the under the current label. Value range: 0 (the lowest confidence) to 100 (the highest confidence). For example, <i>Porn 99</i> indicates that the image is highly likely to be pornographic, while <i>Porn 0</i> indicates that the image is not pornographic. |
| Details | Array of LabelDetailItem | This field is used to return the details of the subtag hit by the categorization model, such as number, hit tag name, and score. Note: this field may return null, indicating that no valid values can be obtained. |

LibDetail

Returns the details of the custom library/blocklist/allowlist

Used by actions: ImageModeration.

| Name | Type | Description |
|---------|---------|---|
| Id | Integer | This field is used to return the ID of the recognized object for easier recognition and distinction. |
| LibId | String | This field is valid only when Label is Custom (custom keyword) . It is used to return the ID of the custom library for easier custom library management and configuration. |
| LibName | String | This field is valid only when Label is Custom (custom keyword) . It is used to return the name of the custom library for easier custom library management and configuration. Note: this field may return null, indicating that no valid values can be obtained. |
| ImageId | String | This field is used to return the ID of the recognized image object for easier file management. |
| Label | String | This field is used to return the maliciousness tag in the detection result. Returned values: Normal : normal; Porn : pornographic; Abuse : abusive; Ad : advertising; Custom : custom type of non-compliant content and other offensive, unsafe, or inappropriate types of content. |
| Tag | String | This field is used to return other custom tags to meet the needs in your customized |

| | | |
|-------|---------|---|
| | | scenarios. It can be skipped if you have no custom needs. Note: this field may return null, indicating that no valid values can be obtained. |
| Score | Integer | This field is used to return the hit score of the model. Value range: 0–100 ; for example, <i>Porn 99</i> indicates that the hit score of the porn tag for the recognized content is 99. |

LibResult

Returns the detailed result of the comparison with the blacklist/allowlist

Used by actions: ImageModeration.

| Name | Type | Description |
|------------|------------------------------------|--|
| Scene | String | This field indicates the scenario recognition result of the model. Default value: Similar. |
| Suggestion | String | This field is used to return the operation suggestion. When you get the determination result, the returned value indicates the operation suggested by the system. We recommend you handle different types of violations and suggestions according to your business needs. Returned values: Block , Review , Pass . |
| Label | String | This field is used to return the maliciousness tag in the detection result. Returned values: Normal : normal; Porn : pornographic; Abuse : abusive; Ad : advertising; Custom : custom type of non-compliant content and other offensive, unsafe, or inappropriate types of content. |
| SubLabel | String | This field is used to return the detection result for a subtag under the maliciousness tag, such as <i>Porn-SexBehavior</i> . Note: this field may return null, indicating that no valid values can be obtained. |
| Score | Integer | This field is used to return the recognition score of the image search model. Value range: 0–100 . It indicates the score for the similarity between the moderated image and the samples in the library . A higher score indicates that the content is more likely to hit a sample in the library of similar images. |
| Details | Array of LibDetail | This field is used to return the detailed result of the comparison with the blacklist/allowlist, such as number, library name, and maliciousness tag. For more information on the returned value, see the description of the LibDetail data structure. Note: this field may return null, indicating that no valid values can be obtained. |

Location

Coordinate

Used by actions: ImageModeration.

| Name | Type | Description |
|--------|-------|--|
| X | Float | This parameter is used to return the pixel position of the abscissa (X) of the top-left corner of the detection frame. It can be combined with other parameters to uniquely determine the size and position of the detection frame. |
| Y | Float | This parameter is used to return the pixel position of the ordinate of the top-left corner (Y) of the detection frame. It can be combined with other parameters to uniquely determine the size and position of the detection frame. |
| Width | Float | This parameter is used to return the width of the detection frame (the length starting from the top-left corner and extending to the right on the X axis). It can be combined with other parameters to uniquely determine the size and position of the detection frame. |
| Height | Float | This parameter is used to return the height of the detection frame (the length starting from the top-left corner and extending down the Y axis). It can be combined with other parameters to uniquely determine the size and position of the detection frame. |
| Rotate | Float | This parameter is used to return the rotation angle of the detection frame . Valid values: 0–360 (degrees) , and the direction is counterclockwise rotation . This parameter can be combined with the <code>X</code> and <code>Y</code> coordinate parameters to uniquely determine the specific position of the detection frame. |

ObjectDetail

Object detection result details. When the detection scenario is an object, advertising logo, or QR code, it represents the tag name, tag value, tag score, and position information of the model detection frame.

Used by actions: ImageModeration.

| Name | Type | Description |
|-------|---------|---|
| Id | Integer | This parameter is used to return the ID of the recognized object for easier recognition and distinction. |
| Name | String | This parameter is used to return the hit object tag. |
| Value | String | This parameter is used to return the value or content of the object tag; for example, when the tag is <i>QR code (QrCode)</i> , this field will be the URL of the recognized QR code. |
| | | |

| | | |
|----------|--------------------------|---|
| Score | Integer | This parameter is used to return the hit score of the object tag. Valid values: 0–100 ; for example, <i>QRCode 99</i> indicates that it is highly likely that the recognized content will hit the QR code tag. |
| Location | Location | This field is used to return the coordinate position (X and Y coordinates of the top-left corner, length, width, and rotation angle) of the object detection frame for quick location of the object information. |
| SubLabel | String | This parameter is used to return the hit object subtag. |

ObjectResult

Returns object detection result details

Used by actions: ImageModeration.

| Name | Type | Description |
|------------|-----------------|--|
| Scene | String | This field is used to return the recognized object scenario result, such as QR code, logo, and image OCR. |
| Suggestion | String | This field is used to return the operation suggestion for the current maliciousness tag. When you get the determination result, the returned value indicates the operation suggested by the system. We recommend you handle different types of violations and suggestions according to your business needs. Returned values: Block , Review , Pass . |
| Label | String | This field is used to return the maliciousness tag in the detection result, which represents the moderation result suggested by the model. We recommend you handle different types of violations and suggestions according to your business needs. Returned values: Normal : normal; Porn : pornographic; Abuse : abusive; Ad : advertising; Custom : custom type of non-compliant content and other offensive, unsafe, or inappropriate types of content. |
| SubLabel | String | This field is used to return the detection result for a subtag under the current maliciousness tag, such as <i>Porn-SexBehavior</i> . |
| Score | Integer | This field is used to return the hit score of a subtag under the current maliciousness tag. Value range: 0–100 ; for example, <i>Porn-SexBehavior 99</i> indicates that the hit score of the <i>Porn-SexBehavior</i> tag for the recognized content is 99. |
| Names | Array of String | This field is used to return the name of the recognized object. Note: this field may return null, indicating that no valid values can be obtained. |

| | | |
|---------|---------------------------------------|---|
| Details | Array of ObjectDetail | <p>This field is used to return the details of the recognized object, such as number, hit tag name, and position coordinates. For more information on the returned value, see the description of the ObjectDetail data structure.</p> <p>Note: this field may return null, indicating that no valid values can be obtained.</p> |
|---------|---------------------------------------|---|

OcrResult

Returns OCR detection result details

Used by actions: ImageModeration.

| Name | Type | Description |
|------------|--|--|
| Scene | String | This field indicates the recognition scenario. Default value: OCR (image OCR). |
| Suggestion | String | <p>This field is used to return the operation suggestion for the maliciousness tag with the highest priority. When you get the determination result, the returned value indicates the operation suggested by the system. We recommend you handle different types of violations and suggestions according to your business needs.</p> <p>Returned values: Block, Review, Pass.</p> |
| Label | String | <p>This field is used to return the maliciousness tag with the highest priority in the OCR detection result, which represents the moderation result suggested by the model. We recommend you handle different types of violations and suggestions according to your business needs.</p> <p>Returned values: Normal: normal; Porn: pornographic; Abuse: abusive; Ad: advertising; Custom: custom type of non-compliant content and other offensive, unsafe, or inappropriate types of content.</p> |
| SubLabel | String | This field is used to return the detection result for a subtag under the current tag (Label), such as <i>Porn-SexBehavior</i> . |
| Score | Integer | This field is used to return the confidence under the current tag (Label). Value range: 0 (the lowest confidence)–100 (the highest confidence), where a higher value indicates that the text is more likely to fall into the category of the current returned tag; for example, <i>Porn 99</i> indicates that the text is highly likely to be pornographic, while <i>Porn 0</i> indicates that the text is not pornographic. |
| Details | Array of OcrTextDetail | <p>This field is used to return the details of the OCR recognition result, such as text content, tag, and recognition frame position.</p> <p>Note: this field may return null, indicating that no valid values can be obtained.</p> |
| Text | String | This field is used to return the text information recognized by OCR. |

OcrTextDetail

This field is used to return the OCR text result details. More text in the image may increase the time it takes the API to return the result.

Used by actions: ImageModeration.

| Name | Type | Description |
|----------|--------------------------|---|
| Text | String | This field is used to return the text content recognized by OCR. Note: OCR can recognize text of up to 5,000 bytes . |
| Label | String | This field is used to return the maliciousness tag in the detection result. Returned values: Normal : normal; Porn : pornographic; Abuse : abusive; Ad : advertising; Custom : custom type of non-compliant content and other offensive, unsafe, or inappropriate types of content. |
| LibId | String | This field is valid only when Label is Custom (custom keyword) . It is used to return the ID of the custom library for easier custom library management and configuration. |
| LibName | String | This field is valid only when Label is Custom (custom keyword) . It is used to return the name of the custom library for easier custom library management and configuration. |
| Keywords | Array of String | This parameter is used to return the hit keyword under the current tag (label). |
| Score | Integer | This parameter is used to return the model hit score of the current maliciousness tag. Value range: 0–100 , where a higher value indicates that the current scenario agrees more with the scenario represented by the maliciousness tag. |
| Location | Location | This parameter is used to return the position (X and Y coordinates of the top-left corner, length, width, and rotation angle) of the OCR detection frame in the image for quick location of the recognized text. |
| Rate | Integer | This parameter is used to return the confidence of the text OCR result. Valid values: 0 (the lowest confidence)–100 (the highest confidence) , where a higher value indicates that it is more likely that the image contains the recognized text; for example, <i>Hello 99</i> indicates that it is highly likely that the text in the OCR recognition frame is "Hello". |
| SubLabel | String | This field is used to return the maliciousness subtag that corresponds to the detection result. |

RecognitionResult

Label of the identification moderation result

Used by actions: ImageModeration.

| Name | Type | Description |
|-------|---|---|
| Label | String | Value: <code>Scene</code> Note: This field may return <code>null</code> , indicating that no valid values can be obtained. |
| Tags | Array of RecognitionTag | Hit tags under the <code>Label</code> Note: This field may return <code>null</code> , indicating that no valid values can be obtained. |

RecognitionTag

Recognition tag information

Used by actions: ImageModeration.

| Name | Type | Description |
|----------|--------------------------|--|
| Name | String | Tag name Note: This field may return <code>null</code> , indicating that no valid values can be obtained. |
| Score | Integer | Confidence score. Value: 1 to 100. Note: This field may return <code>null</code> , indicating that no valid values can be obtained. |
| Location | Location | Location information. It returns 0 if there is not location information. Note: This field may return <code>null</code> , indicating that no valid values can be obtained. |

User

Indicates the information of the business user's account

Used by actions: ImageModeration.

| Name | Type | Required | Description |
|--------|--------|----------|---|
| UserId | String | No | This field indicates the business user ID. After it is specified, the |

| | | | |
|-------------|---------|----|---|
| | | | system can optimize the moderation result according to the violation history to facilitate determination when a suspicious violation risk exists. |
| Nickname | String | No | This field indicates the nickname of the business user's account. |
| AccountType | String | No | This field indicates the account type of the business user ID. This field can be used together with the ID parameter (UserId) to uniquely identify the account. |
| Gender | Integer | No | This field indicates the gender of the business user's account. Valid values: 0 (default value): unknown; 1 (male); 2 (female). |
| Age | Integer | No | This field indicates the age of the business user's account. Valid values: integers between 0 (default value, which indicates unknown) and custom age limit . |
| Level | Integer | No | This field indicates the level of the business user's account. Valid values: 0 (default value): unknown; 1 : low level; 2 : medium level; 3 : high level. Currently, the level is not customizable . |
| Phone | String | No | This field indicates the mobile number of the business user's account. It supports recording mobile numbers across the world. Note: you need to use a consistent mobile number format, such as area code format (086/+86). |
| Desc | String | No | This field indicates the profile of the business user. It can contain up to 5,000 letters and special symbols . |
| HeadUrl | String | No | This field indicates the access URL of the business user's profile photo in PNG, JPG, JPEG, BMP, GIF, or WEBP format. Note: the profile photo cannot exceed 5 MB in size. A resolution of 256x256 or higher is recommended. The image download time should be limited to 3 seconds; otherwise, a download timeout will be returned. |

Error Codes

Last updated : 2024-11-25 17:03:40

Feature Description

If there is an Error field in the response, it means that the API call failed. For example:

```
{
  "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"
  }
}
```

Code in Error indicates the error code, and Message indicates the specific information of the error.

Error Code List

Common Error Codes

| Error Code | Description |
|----------------------------------|--|
| ActionOffline | This API has been deprecated. |
| AuthFailure.InvalidAuthorization | <code>Authorization</code> in the request header is invalid. |
| AuthFailure.InvalidSecretId | Invalid key (not a TencentCloud API key type). |
| AuthFailure.MFAFailure | MFA failed. |
| AuthFailure.SecretIdNotFound | Key does not exist. Check if the key has been deleted or disabled in the console, and if not, check if the key is correctly entered. Note that whitespaces should not exist before or after the key. |
| AuthFailure.SignatureExpire | Signature expired. Timestamp and server time cannot differ by more than five minutes. Please |

| | |
|---|---|
| | ensure your current local time matches the standard time. |
| AuthFailure.SignatureFailure | Invalid signature. Signature calculation error. Please ensure you've followed the signature calculation process described in the Signature API documentation. |
| AuthFailure.TokenFailure | Token error. |
| AuthFailure.UnauthorizedOperation | The request is not authorized. For more information, see the CAM documentation. |
| DryRunOperation | DryRun Operation. It means that the request would have succeeded, but the DryRun parameter was used. |
| FailedOperation | Operation failed. |
| InternalServerError | Internal error. |
| InvalidAction | The API does not exist. |
| InvalidParameter | Incorrect parameter. |
| InvalidParameterValue | Invalid parameter value. |
| InvalidRequest | The multipart format of the request body is incorrect. |
| IpInBlacklist | Your IP is in uin IP blacklist. |
| IpNotInWhitelist | Your IP is not in uin IP whitelist. |
| LimitExceeded | Quota limit exceeded. |
| MissingParameter | A parameter is missing. |
| NoSuchProduct | The product does not exist. |
| NoSuchVersion | The API version does not exist. |
| RequestLimitExceeded | The number of requests exceeds the frequency limit. |
| RequestLimitExceeded.GlobalRegionUinLimitExceeded | Uin exceeds the frequency limit. |
| RequestLimitExceeded.IPLimitExceeded | The number of ip requests exceeds the frequency limit. |
| RequestLimitExceeded.UinLimitExceeded | The number of uin requests exceeds the frequency |

| | |
|---------------------------|---|
| | limit. |
| RequestSizeLimitExceeded | The request size exceeds the upper limit. |
| ResourceInUse | Resource is in use. |
| ResourceInsufficient | Insufficient resource. |
| ResourceNotFound | The resource does not exist. |
| ResourceUnavailable | Resource is unavailable. |
| ResponseSizeLimitExceeded | The response size exceeds the upper limit. |
| ServiceUnavailable | Service is unavailable now. |
| UnauthorizedOperation | Unauthorized operation. |
| UnknownParameter | Unknown parameter. |
| UnsupportedOperation | Unsupported operation. |
| UnsupportedProtocol | HTTP(S) request protocol error; only GET and POST requests are supported. |
| UnsupportedRegion | API does not support the requested region. |

Service Error Codes

| Error Code | Description |
|---|--|
| InternalServerError.InternalError | An internal error occurred. |
| InvalidParameter.ImageAspectRatioTooLarge | |
| InvalidParameter.ImageDataTooSmall | |
| InvalidParameter.ImageSizeTooSmall | The image resolution is too low. |
| InvalidParameter.InvalidImageContent | The image content is incorrect. |
| InvalidParameter.InvalidParameter | The parameter is invalid. |
| InvalidParameterValue.EmptyImageContent | The image content parameter is empty. |
| InvalidParameterValue.ImageSizeTooSmall | The image resolution is too low. |
| InvalidParameterValue.InvalidContent | <code>FileContent</code> and <code>FileUrl</code> are empty. |

| | |
|--|--|
| InvalidParameterValue.InvalidDataId | The <code>DataId</code> format is incorrect. |
| InvalidParameterValue.InvalidFileContentSize | The image file content size is exceptional. |
| InvalidParameterValue.InvalidImageContent | The image content is incorrect. |
| InvalidParameterValue.InvalidParameter | The parameter value is incorrect. |
| OperationDenied | The operation was denied. |
| ResourceUnavailable.ImageDownloadError | Image file download failed. |
| ResourceUnavailable.InvalidImageContent | The image resource is incorrect. |
| ResourceUnavailable.ModelCallFailed | |
| ResourcesSoldOut | The resources have been sold out. |
| UnauthorizedOperation.Unauthorized | Operation not authorized/No valid package/The account is overdue |