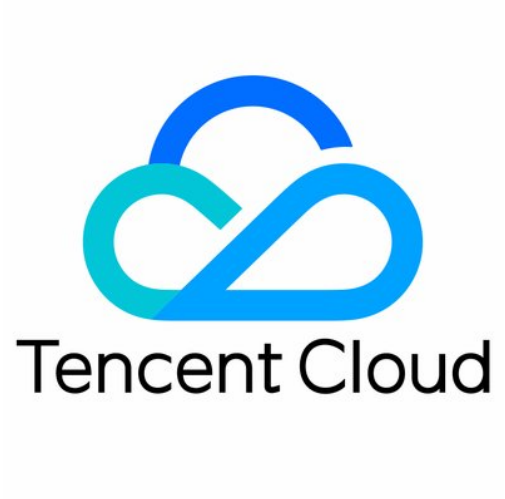
CloudAudit

API Documentation

Product Documentation



Contents

API Documentation History

Last updated : 2022-03-22 16:19:36

Release 5

Release time: 2022-03-16 10:50:28

Release updates:

Improvement to existing documentation.

New APIs:

- CreateAuditTrack
- DeleteAuditTrack
- ModifyAuditTrack

New data structures:

- AttributeKeyDetail
- AuditSummary
- CmqRegionInfo
- CosRegionInfo

Release 4

Release time: 2022-02-25 14:26:43

Release updates:

Improvement to existing documentation.

New APIs:

- DescribeAuditTracks

Release 3

Release time: 2021-11-24 17:13:52

Release updates:

Improvement to existing documentation.

Modified APIs:

- DescribeEvents
  - New output parameters:TotalCount

Release 2

Release time: 2021-10-21 16:13:08

Release updates:

Improvement to existing documentation.

New APIs:

- DescribeEvents

Deleted APIs:

- CreateRecorder
- DeleteRecorder
- DescribeDiscoveredResource
- DescribeRecorder
- GetConfigurationItems
- ListDiscoveredResources
- ListSupportResourceTypes
- UpdateRecorder

New data structures:

- Event
- LookupAttribute
- Resource

Deleted data structures:

- ConfigurationItems
- RecordResourceType

- RelatedEvent
- Resources
- SupportResourceType

Existing Release

Release time: 2020-12-17 16:59:56

Existing APIs/data structures are as follows:

Improvement to existing documentation.

Existing APIs:

- CreateRecorder
- DeleteRecorder
- DescribeDiscoveredResource
- DescribeRecorder
- GetConfigurationItems
- ListDiscoveredResources
- ListSupportResourceTypes
- UpdateRecorder

Existing data structures:

- ConfigurationItems
- RecordResourceType
- RelatedEvent
- Resources
- SupportResourceType

Introduction

Last updated : 2021-03-25 10:42:29

Tencent Cloud CloudAudit (CA) allows you to obtain your Tencent Cloud account’s API call records. This includes API calls via the console, SDKs, CLI and other Tencent Cloud services. CA monitors all deployments within Tencent Cloud. With CA, you can trace the user, source IP address and time of all API calls. You can set multiple tracking sets to store different logs, which can be enabled and disabled at any time.

⚠ Note :

All CA APIs in this section have been upgraded to API 3.0. Future CAfeatures will also be added here. We recommend using API 3.0.

API Category

Last updated : 2022-03-18 16:49:49

CloudAudit APIs

API Name	Feature
DescribeEvents	Queries CloudAudit logs
DescribeAuditTracks	Queries the CloudAudit tracking set list
ModifyAuditTrack	Modifies CloudAudit tracking set
DeleteAuditTrack	Deletes CloudAudit tracking set
CreateAuditTrack	Creating CloudAudit tracking set

Calling Method
Request Structure

Last updated : 2021-10-21 16:17:54

1. Service Address

The API supports access from either a nearby region (at cloudaudit.tencentcloudapi.com) or a specified region (at cloudaudit.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 "cloudaudit.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)	cloudaudit.tencentcloudapi.com
South China (Guangzhou)	cloudaudit.ap-guangzhou.tencentcloudapi.com
East China (Shanghai)	cloudaudit.ap-shanghai.tencentcloudapi.com
North China (Beijing)	cloudaudit.ap-beijing.tencentcloudapi.com
Southwest China (Chengdu)	cloudaudit.ap-chengdu.tencentcloudapi.com
Southwest China (Chongqing)	cloudaudit.ap-chongqing.tencentcloudapi.com
Hong Kong, Macao, Taiwan (Hong Kong, China)	cloudaudit.ap-hongkong.tencentcloudapi.com

Southeast Asia (Singapore)	cloudaudit.ap-singapore.tencentcloudapi.com
Southeast Asia (Bangkok)	cloudaudit.ap-bangkok.tencentcloudapi.com
South Asia (Mumbai)	cloudaudit.ap-mumbai.tencentcloudapi.com
Northeast Asia (Seoul)	cloudaudit.ap-seoul.tencentcloudapi.com
Northeast Asia (Tokyo)	cloudaudit.ap-tokyo.tencentcloudapi.com
U.S. East Coast (Virginia)	cloudaudit.na-ashburn.tencentcloudapi.com
U.S. West Coast (Silicon Valley)	cloudaudit.na-siliconvalley.tencentcloudapi.com
North America (Toronto)	cloudaudit.na-toronto.tencentcloudapi.com
Europe (Frankfurt)	cloudaudit.eu-frankfurt.tencentcloudapi.com
Europe (Moscow)	cloudaudit.eu-moscow.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 Parameters

Last updated : 2021-10-21 16:17:54

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 spe description of common parameter `Action` in the input pa documentation. For example, the API for querying the CVM `DescribeInstances` .
X-TC-Region	String	Yes	Region parameter, which is used to identify the region to w work with belongs. For values supported for an API, see th parameter `Region` in the input parameters in related API d parameter is not required for some APIs (which will be indi documentation), and will not take effect even it is passed.
X-TC-Timestamp	Integer	Yes	The current UNIX timestamp that records the time when th for example, 1529223702. Note: If the difference between the server time is greater than 5 minutes, a signature expi
X-TC-Version	String	Yes	API version of the action. For the valid values, see the desc parameter `Version` in the API documentation. For exampl 2017-03-12.
Authorization	String	Yes	The HTTP authentication request header, for example: TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service SignedHeaders=content-type;host, Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2 Here: - TC3-HMAC-SHA256: Signature method, currently fixed as - Credential: Signature credential; AKIDEXAMPLE is the Sec

			time, and this value must match the value of X-TC-Timesta in UTC time format; service is the name of the product/ser domain name prefix. For example, a domain name cvm.te the CVM product and the value would be cvm; - SignedHeaders: The headers that contains the authentic type and host are the required headers; - Signature: Signature digest.
X-TC-Token	String	No	The token used for a temporary certificate. It must be use can obtain the temporary key and token by calling a CAM 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/?Limit=10&Offset=0

Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE/2018-10-09/cvm/tc3
_request, SignedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50
567dba16c6ccf174c474
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=AKIDEXAMPLE/2018-05-30/cvm/tc3_request, SignedHeaders=c
ontent-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=AKIDEXAMPLE/2018-05-30/cvm/tc3_request, SignedHeaders=c
ontent-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 `Action` in the input parameters in related API documentation. For example, the API for querying the CVM instance list is `DescribeInstances`.
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 `Region` 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 `Timestamp` 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 `Version` 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=37ac2f4fde00b0ac9bd9eadeb459b1bbee224158d66e7ae5fcadb70b2d181d02&Region=ap-guangzhou&Nonce=23823223&SecretId=AKIDEXAMPLE

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=AKIDEXAMPLE
```

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
South China (Guangzhou)	ap-guangzhou
Hong Kong/Macao/Taiwan (Hong Kong, China)	ap-hongkong
Northeast Asia Pacific (Seoul)	ap-seoul
Southeast Asia Pacific (Singapore)	ap-singapore
Northeast Asia Pacific (Tokyo)	ap-tokyo
Europe (Frankfurt)	eu-frankfurt
Europe (Moscow)	eu-moscow

Signature Algorithm v3

Last updated : 2020-10-22 09:34:14

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 `AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******` and `Gu5t9xGARNpq86cd98joQYCN3*******`, 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=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host, Signature=c492e8e41437e97a620b728c301bb8d17e7dc0c17eeabce80c20cd70fc3a78ff" ¥
-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 `POST`.
CanonicalURI	URI parameter. Slash ("/") is used for API 3.0.
CanonicalQueryString	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: Limit=10&Offset=0。 Note: `CanonicalQueryString` 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.
CanonicalHeaders	Header information for signature calculation, including at least two headers of `host` and `content-type`. Custom headers can be added to participate in the signature process to improve the uniqueness and security of the request. Concatenation rules: 1. 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 key:value\n format; 2. If there are multiple headers, they should be sorted in ASCII ascending order by the header keys (lowercase). The calculation result in this example is `content-type:application/json; charset=utf-8¥nhost:cvm.tencentcloudapi.com¥n`. Note: `content-type` 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 sever will return an error indicating that signature verification failed.
SignedHeaders	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). `Content-type` and `host` are required headers. Concatenation rules: 1. Both the key and value of the header should be converted to lowercase; 2. If there are multiple headers, they should be sorted in ASCII ascending order by the header keys (lowercase) and separated by semicolons (;). The value in this example is `content-type;host`
HashedRequestPayload	Hash value of the request payload (i.e., the body, such as `{"Limit": 1, "Filters": [{"Values": ["unnamed"], "Name": "instance-name"}]}` in this example). The pseudocode for calculation is Lowercase(HexEncode(Hash.SHA256(RequestPayload))) by SHA256 hashing the payload of the HTTP request, performing hexadecimal encoding, and finally converting the encoded string to lowercase letters. For GET requests, `RequestPayload` is always an empty 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 the 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 the 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 `2019-02-25/cvm/tc3_request`.
HashedCanonicalRequest	Hash value of the CanonicalRequest string concatenated in the steps above. The pseudocode for calculation is Lowercase(HexEncode(Hash.SHA256(CanonicalRequest))). The calculation result in this example is `2815843035062fffda5fd6f2a44ea8a34818b0dc46f024b8b3786976a3adda7a`.

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
2815843035062fffda5fd6f2a44ea8a34818b0dc46f024b8b3786976a3adda7a
```

3. Calculating the Signature

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

```
SecretKey = "Gu5t9xGARNpq86cd98joQYCN3*******"
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., `Gu5t9xGARNpq86cd98joQYCN3*******`.
Date	The Date field information in `Credential`, such as `2019-02-25` in this example.
Service	Value in the Service field in `Credential`, such as `cvm` 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 `TC3-HMAC-SHA256`.
SecretId	The SecretId in the key pair, i.e., `AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******`.
CredentialScope	Credential scope (see above). The calculation result in this example is `2019-02-25/cvm/tc3_request`.
SignedHeaders	Header information for signature calculation (see above), such as `content-type;host` in this example.
Signature	Signature value. The calculation result in this example is `c492e8e41437e97a620b728c301bb8d17e7dc0c17eeabce80c20cd70fc3a78ff`.

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

```
TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host, Signature=c492e8e41437e97a620b728c301bb8d17e7dc0c17eeabce80c20cd70fc3a78ff
```

The following example shows a finished authorization header:

```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host, Signature=c492e8e41437e97a620b728c301bb8d17e7dc0c17eeabce80c20cd70fc3a78ff
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=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******&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 = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******";
private final static String SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3*******";
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-n
ame¥"}]}";
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" + hashedCanoni
calRequest;
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)).toLower
Case();
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 = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3*******"

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": [{"Name": "instance-name", "Values": ["unnamed"]}]}

# ************* 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 := "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******"
secretKey := "Gu5t9xGARNpq86cd98joQYCN3*******"
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 = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******";
$secretKey = "Gu5t9xGARNpq86cd98joQYCN3*******";
$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 = 'AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******'
secret_key = 'Gu5t9xGARNpq86cd98joQYCN3*******'

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"
signed_headers = 'content-type;host'
# params = { 'Limit' => 1, 'Filters' => [{ 'Name' => 'instance-name', 'Values' => ['unnamed'] }]
}
# payload = JSON.generate(params, { 'ascii_only' => true, 'space' => ' ' })
# json will generate in random order, to get specified result in example, we hard-code it here.
payload = '{"Limit": 1, "Filters": [{"Values": ["unnamed"], "Name": "instance-name"}]}'
hashed_request_payload = Digest::SHA256.hexdigest(payload)
canonical_request = [
http_request_method,
canonical_uri,
canonical_querystring,
canonical_headers,
signed_headers,
hashed_request_payload,
].join("¥n")

puts canonical_request

# ************* Step 2: Concatenate the string to sign *************
credential_scope = date + '/' + service + '/' + 'tc3_request'
hashed_request_payload = Digest::SHA256.hexdigest(canonical_request)
string_to_sign = [
algorithm,
timestamp.to_s,
credential_scope,
hashed_request_payload,
].join("¥n")
puts string_to_sign

# ************* Step 3: Calculate the Signature *************
digest = OpenSSL::Digest.new('sha256')
secret_date = OpenSSL::HMAC.digest(digest, 'TC3' + secret_key, date)
secret_service = OpenSSL::HMAC.digest(digest, secret_date, service)
secret_signing = OpenSSL::HMAC.digest(digest, secret_service, 'tc3_request')
signature = OpenSSL::HMAC.hexdigest(digest, secret_signing, string_to_sign)
puts signature

# ************* Step 4: Concatenate the Authorization *************
authorization = "#{algorithm} Credential=#{secret_id}/#{credential_scope}, SignedHeaders=#{signed
_headers}, Signature=#{signature}"
puts authorization

puts 'curl -X POST ' + endpoint ¥
+ ' -H "Authorization: ' + authorization + '"' ¥
+ ' -H "Content-Type: application/json; charset=utf-8"' ¥
```

```
+ ' -H "Host: ' + host + '"' ¥
+ ' -H "X-TC-Action: ' + action + '"' ¥
+ ' -H "X-TC-Timestamp: ' + timestamp.to_s + '"' ¥
+ ' -H "X-TC-Version: ' + version + '"' ¥
+ ' -H "X-TC-Region: ' + region + '"' ¥
+ " -d '" + payload + "'"
```

DotNet

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

public class Application
{
public static string SHA256Hex(string s)
{
using (SHA256 algo = SHA256.Create())
{
byte[] hashbytes = algo.ComputeHash(Encoding.UTF8.GetBytes(s));
StringBuilder builder = new StringBuilder();
for (int i = 0; i < hashbytes.Length; ++i)
{
builder.Append(hashbytes[i].ToString("x2"));
}
return builder.ToString();
}
}
public static byte[] HmacSHA256(byte[] key, byte[] msg)
{
using (HMACSHA256 mac = new HMACSHA256(key))
{
return mac.ComputeHash(msg);
}
}

public static Dictionary<String, String> BuildHeaders(string secretid,
string secretkey, string service, string endpoint, string region,
string action, string version, DateTime date, string requestPayload)
{
string datestr = date.ToString("yyyy-MM-dd");
DateTime startTime = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc);
long requestTimestamp = (long)Math.Round((date - startTime).TotalMilliseconds, MidpointRounding.A
wayFromZero) / 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:" + endpoin
t + "¥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" + credentialScope +
"¥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_request"));
byte[] signatureBytes = HmacSHA256(secretSigning, Encoding.UTF8.GetBytes(stringToSign));
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 = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******";
string SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3*******";

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 fol
lowing 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¥": [¥"¥¥u672a¥¥u547d¥¥u540d¥"],
¥"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 = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******"
const SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3*******"

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-na
me¥"}]}"

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++)
{
snprintf(buf, sizeof(buf), "%02x", hash[i]);
NewString = NewString + buf;
}
return NewString;
}
string HmacSha256(const string &key, const string &input)
{
unsigned char hash[32];

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

```
#endif

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

#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX_cleanup(h);
#else
HMAC_CTX_free(h);
#endif

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

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

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

int main()
{
string SECRET_ID = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******";
string SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3*******";

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-n
ame¥"}]}";
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" + hashe
dCanonicalRequest;
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 + "/" + credentialScope + ", "
+ "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 Algorithm

Last updated : 2021-11-15 17:18:13

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: `AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******`
- SecretKey: `Gu5t9xGARNpq86cd98joQYCN3*******`

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	`AKIDz8krbsJ5yKBZQpn74WFkmLPx3******`
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' : 'AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******',
'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=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******&Timestamp=1465185768&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&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******&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 = 'Gu5t9xGARNpq86cd98joQYCN3*******';
$srcStr = 'GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******&Timestamp=1465185768&Version=2017-03-12';
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));
echo $signStr;
```

The final signature is:

```
zmmjn35mikh6pM3V7sUEuX4wyYM=
```

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 zmmjn35mikh6pM3V7sUEuX4wyYM=, the final signature string request parameter (Signature) is zmmjn35mikh6pM3V7sUEuX4wyYM%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=AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******&Signature=zmmjn35mikh6pM3V7sUEuX4wyYM%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 val
```

```
ue 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 enables automatic sort
ing
// A random number should be used when actually calling, for example: params.put("Nonce", new Ran
dom().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", "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******"); // 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), "Gu5t9xGARNpq86cd98joQYCN3*******", "HmacSH
A1")); // 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 = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3*******"
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 := "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******"
secretKey := "Gu5t9xGARNpq86cd98joQYCN3*******"
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 = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******";
$secretKey = "Gu5t9xGARNpq86cd98joQYCN3*******";
$param["Nonce"] = 11886;//rand();
$param["Timestamp"] = 1465185768;//time();
$param["Region"] = "ap-guangzhou";
$param["SecretId"] = $secretId;
$param["Version"] = "2017-03-12";
$param["Action"] = "DescribeInstances";
$param["InstanceIds.0"] = "ins-09dx96dg";
$param["Limit"] = 20;
$param["Offset"] = 0;
ksort($param);
$signStr = "GETcvm.tencentcloudapi.com/?";
foreach ( $param as $key => $value ) {
$signStr = $signStr . $key . "=" . $value . "&";
}
$signStr = substr($signStr, 0, -1);
$signature = base64_encode(hash_hmac("sha1", $signStr, $secretKey, true));
echo $signature.PHP_EOL;
// need to install and enable curl extension in php.ini
// $param["Signature"] = $signature;
```

```
// $url = "https://cvm.tencentcloudapi.com/?".http_build_query($param);
// echo $url.PHP_EOL;
// $ch = curl_init();
// curl_setopt($ch, CURLOPT_URL, $url);
// $output = curl_exec($ch);
// curl_close($ch);
// echo json_decode($output);
```

Ruby

```
# -*- coding: UTF-8 -*-
# require ruby>=2.3.0
require 'time'
require 'openssl'
require 'base64'
secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3*******"
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 req
uestMethod, 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 = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******";
string SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3*******";
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>(param, StringComp
arer.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(sigOutParam);
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 = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3*******"
const SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3*******"
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 : 2020-02-15 11:44:32

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 corr
ect."
},
"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
AuthFailure.InvalidSecretId	Invalid key (not a TencentCloud API key type).
AuthFailure.MFAFailure	MFA failed.
AuthFailure.SecretIdNotFound	The key does not exist.
AuthFailure.SignatureExpire	Signature expired.
AuthFailure.SignatureFailure	Signature error.
AuthFailure.TokenFailure	Token error.
AuthFailure.UnauthorizedOperation	The request does not have CAM authorization.
DryRunOperation	DryRun Operation. It means that the request would have succeeded, but the DryRun parameter was used.
FailedOperation	Operation failed.
InternalError	Internal error.
InvalidAction	The API does not exist.
InvalidParameter	Incorrect parameter.
InvalidParameterValue	Invalid parameter value.
LimitExceeded	Quota limit exceeded.

Error Code	Description
MissingParameter	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.

CloudAudit APIs DescribeEvents

Last updated : 2022-02-25 14:27:45

1. API Description

Domain name for API request: cloudaudit.tencentcloudapi.com.

This API is used to query CloudAudit logs.

A maximum of 20 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.
This document describes the parameters for Signature V1. It's recommended to use the V3 signature, which provides higher security. Note that for Signature V3, the common parameters need to be placed in the HTTP Header. See details.

Parameter Name	Required	Type	Description
Action	Yes	String	Common parameter. The value used for this API: DescribeEvents.
Version	Yes	String	Common parameter. The value used for this API: 2019-03-19.
Region	Yes	String	Common parameter. For more information, please see the list of regions supported by the product.
StartTime	Yes	Integer	Start timestamp in seconds (cannot be 90 days after the current time).

EndTime	Yes	Integer	End timestamp in seconds (the time range for query is less than 30 days).
NextToken	No	Integer	Credential for viewing more logs.
MaxResults	No	Integer	Max number of returned logs (up to 50).
LookupAttributes.N	No	Array of LookupAttribute	Search criterion. Valid values: RequestId, EventName, ActionType (write/read), PrincipalId (sub-account), ResourceType, ResourceName, AccessKeyId, SensitiveAction, ApiErrorCode, and CamErrorCode.
IsReturnLocation	No	Integer	Whether to return the IP location. `1` : yes, `0` : no.

3. Output Parameters

Parameter Name	Type	Description
ListOver	Boolean	Whether the logset ends.
NextToken	Integer	Credential for viewing more logs.
Events	Array of Event	Logset. Note: `null` may be returned for this field, indicating that no valid values can be obtained.
TotalCount	Integer	Total number of events. Note: this field may return `null` , indicating that no valid values can be obtained.
RequestId	String	The unique request ID, which is returned for each request. RequestId is required for locating a problem.

4. Example

Example1 Querying CloudAudit logs

Input Example

```
POST / HTTP/1.1
Host: cloudaudit.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DescribeEvents
<Common request parameters>

{
"StartTime": 1610613170,
"EndTime": 1610699570,
"MaxResults": 1
}
```

Output Example


```
{
"Response": {
"ListOver": false,
"TotalCount": 1,
"Events": [
{
"CloudAuditEvent": "{¥"userIdentity¥":{¥"principalId¥":¥"100000000000¥",¥"accountId¥":¥"100000000000¥",¥"secretId¥":¥"xxx¥",¥"type¥":¥"Root¥",¥"userName¥":¥"root¥",¥"sessionContext¥":¥"{¥¥¥"token¥¥¥": ¥¥¥"xxx¥¥¥", ¥¥¥"userIp¥¥¥": ¥¥¥"163.177.68.30¥¥¥", ¥¥¥"uin¥¥¥": 100000000000, ¥¥¥"ownerUin¥¥¥": 100000000000, ¥¥¥"appId¥¥¥": 1000000000, ¥¥¥"expireTime¥¥¥": ¥¥¥"2021-01-15 17:35:55¥¥¥", ¥¥¥"mfa¥¥¥": 0, ¥¥¥"mfaExpireTime¥¥¥": ¥¥¥"0000-00-00 00:00:00¥¥¥", ¥¥¥"interfaceName¥¥¥": ¥¥¥"¥¥¥", ¥¥¥"hasPolicyFilter¥¥¥": 0, ¥¥¥"policyFilter¥¥¥": ¥¥¥"¥¥¥", ¥¥¥"extraInfo¥¥¥": ¥¥¥"¥¥¥"}¥"},¥"@timestamp¥":¥"2021-01-15T07:35:59.115042¥",¥"onlyRecordNotSeen¥":¥"0¥",¥"eventRegion¥":¥"ap-guangzhou¥",¥"eventVersion¥":2,¥"errorCode¥":¥"0¥",¥"errorMessage¥":¥"permission verify¥",¥"requestID¥":¥"c8c04477-eb9e-4703-84ae-f8758c6084ff¥",¥"eventID¥":¥"c8c04477-eb9e-4703-84ae-f8758c6084ff¥",¥"apiVersion¥":¥"3.0¥",¥"eventType¥":¥"ConsoleCall¥",¥"actionType¥":¥"Read¥",¥"authMode¥":¥"0¥",¥"isRisk¥":¥"0¥",¥"ruleId¥":¥"0¥",¥"httpMethod¥":¥"POST¥",¥"apiErrorCode¥":¥"0¥",¥"apiErrorMessage¥":¥"¥",¥"userAgent¥":¥"SDK_NODEJS_0.2.1¥",¥"eventTime¥":1610696155,¥"updateEsTime¥":1610696161641644206,¥"sensitiveAction¥":¥"¥",¥"eventPlatform¥":¥"0¥",¥"sourceIPAddress¥":¥"9.83.55.32¥",¥"resourceType¥":¥"cloudaudit¥",¥"eventName¥":¥"LookUpEvents¥",¥"eventSource¥":¥"cloudaudit.ap-chongqing.api.tencentyun.com¥",¥"requestParameters¥":¥"{¥¥¥"Region¥¥¥": ¥¥¥"ap-guangzhou¥¥¥", ¥¥¥"SecretId¥¥¥": ¥¥¥"xxx¥¥¥", ¥¥¥"Timestamp¥¥¥": ¥¥¥"1610696155¥¥¥", ¥¥¥"Nonce¥¥¥": ¥¥¥"11289¥¥¥", ¥¥¥"RequestClient¥¥¥": ¥¥¥"SDK_NODEJS_0.2.1¥¥¥", ¥¥¥"StartTime¥¥¥": ¥¥¥"1610121600¥¥¥", ¥¥¥"EndTime¥¥¥": ¥¥¥"1610726399¥¥¥", ¥¥¥"MaxResults¥¥¥": ¥¥¥"20¥¥¥", ¥¥¥"Mode¥¥¥": ¥¥¥"standard¥¥¥", ¥¥¥"Version¥¥¥": ¥¥¥"2019-03-19¥¥¥", ¥¥¥"Action¥¥¥": ¥¥¥"LookUpEvents¥¥¥", ¥¥¥"RequestOperator¥¥¥": ¥¥¥"100000000000¥¥¥", ¥¥¥"Token¥¥¥": ¥¥¥"xxx¥¥¥", ¥¥¥"RequestSource¥¥¥": ¥¥¥"MC¥¥¥", ¥¥¥"seqId¥¥¥": ¥¥¥"1be35142-f784-64d4-4502-a1250702edcd¥¥¥"}¥",¥"resources¥":¥"[¥¥¥"*¥¥¥"]¥",¥"resourceName¥":¥"¥",¥"cloudapi¥":1,¥"auth¥":1,¥"signature¥":0}",
"EventName": "LookUpEvents",
"EventTime": 1610696155,
"SecretId": "xxx",
"ErrorCode": "0",
```

```
"RequestID": "c8c04477-eb9e-4703-84ae-f8758c6084ff",
"SourceIPAddress": "9.83.55.32",
"EventSource": "cloudaudit.ap-chongqing.api.tencentyun.com",
"EventRegion": "ap-guangzhou",
"Resources": {
"ResourceName": "",
"ResourceType": "cloudaudit"
},
"Username": "root",
"ResourceTypeCn": "CloudAudit",
"EventNameCn": "",
"ResourceRegion": ""
}
],
"NextToken": 16106961641644206,
"RequestId": "2d4a7fba-bba8-452e-a99e-ccf11fdaa583"
}
}
```


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 NodeJS
- 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
InternalError	Internal error.
InvalidParameter	Parameter error.

DescribeAuditTracks

Last updated : 2022-02-25 14:27:45

1. API Description

Domain name for API request: cloudaudit.tencentcloudapi.com.

This API is used to query the CloudAudit tracking set list.

A maximum of 20 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.
This document describes the parameters for Signature V1. It's recommended to use the V3 signature, which provides higher security. Note that for Signature V3, the common parameters need to be placed in the HTTP Header. See details.

Parameter Name	Required	Type	Description
Action	Yes	String	Common parameter. The value used for this API: DescribeAuditTracks.
Version	Yes	String	Common parameter. The value used for this API: 2019-03-19.
Region	Yes	String	Common parameter. For more information, please see the list of regions supported by the product.

3. Output Parameters

Parameter Name	Type	Description
RequestId	String	The unique request ID, which is returned for each request. RequestId is required for locating a problem.

4. Example

Example1 Querying CloudAudit tracking set list

Input Example

```
POST / HTTP/1.1
Host: cloudaudit.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DescribeAuditTracks
<Common request parameters>

{}
```

Output Example

```
{
"Response": {
"RequestId": "2d4a7fba-bba8-452e-a99e-ccf11fdaa583"
}
}
```

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 NodeJS

- 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
InternalError	Internal error.
InvalidParameter	Parameter error.
InvalidParameterValue.AliasAlreadyExists	The alias already exists.
LimitExceeded.OverAmount	The maximum number of tracking sets has been exceeded.

ModifyAuditTrack

Last updated : 2022-03-16 10:53:23

1. API Description

Domain name for API request: cloudaudit.tencentcloudapi.com.

This API is used to modify a CloudAudit tracking set.

A maximum of 20 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.
This document describes the parameters for Signature V1. It's recommended to use the V3 signature, which provides higher security. Note that for Signature V3, the common parameters need to be placed in the HTTP Header. See details.

Parameter Name	Required	Type	Description
Action	Yes	String	Common parameter. The value used for this API: ModifyAuditTrack.
Version	Yes	String	Common parameter. The value used for this API: 2019-03-19.
Region	Yes	String	Common parameter. For more information, please see the list of regions supported by the product.

3. Output Parameters

Parameter Name	Type	Description
RequestId	String	The unique request ID, which is returned for each request. RequestId is required for locating a problem.

4. Example

Example1 Modifying CloudAudit tracking set

Input Example

```
POST / HTTP/1.1
Host: cloudaudit.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: ModifyAuditTrack
<Common request parameters>

{}
```

Output Example

```
{
"Response": {
"RequestId": "2d4a7fba-bba8-452e-a99e-ccf11fdaa583"
}
}
```

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 NodeJS

- 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
InternalError	Internal error.
InvalidParameter	Parameter error.
InvalidParameterValue.AliasAlreadyExists	The alias already exists.
LimitExceeded.OverAmount	The maximum number of tracking sets has been exceeded.

DeleteAuditTrack

Last updated : 2022-03-16 10:53:25

1. API Description

Domain name for API request: cloudaudit.tencentcloudapi.com.

This API is used to delete a CloudAudit tracking set.

A maximum of 20 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.
This document describes the parameters for Signature V1. It's recommended to use the V3 signature, which provides higher security. Note that for Signature V3, the common parameters need to be placed in the HTTP Header. See details.

Parameter Name	Required	Type	Description
Action	Yes	String	Common parameter. The value used for this API: DeleteAuditTrack.
Version	Yes	String	Common parameter. The value used for this API: 2019-03-19.
Region	Yes	String	Common parameter. For more information, please see the list of regions supported by the product.

3. Output Parameters

Parameter Name	Type	Description
RequestId	String	The unique request ID, which is returned for each request. RequestId is required for locating a problem.

4. Example

Example1 Deleting CloudAudit tracking set

Input Example

```
POST / HTTP/1.1
Host: cloudaudit.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DeleteAuditTrack
<Common request parameters>

{}
```

Output Example

```
{
"Response": {
"RequestId": "2d4a7fba-bba8-452e-a99e-ccf11fdaa583"
}
}
```

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 NodeJS

- 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
InternalError	Internal error.
InvalidParameter	Parameter error.
InvalidParameterValue.AliasAlreadyExists	The alias already exists.
LimitExceeded.OverAmount	The maximum number of tracking sets has been exceeded.

CreateAuditTrack

Last updated : 2022-03-16 10:53:26

1. API Description

Domain name for API request: cloudaudit.tencentcloudapi.com.

This API is used to create a tracking set.

A maximum of 20 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.
This document describes the parameters for Signature V1. It's recommended to use the V3 signature, which provides higher security. Note that for Signature V3, the common parameters need to be placed in the HTTP Header. See details.

Parameter Name	Required	Type	Description
Action	Yes	String	Common parameter. The value used for this API: CreateAuditTrack.
Version	Yes	String	Common parameter. The value used for this API: 2019-03-19.
Region	Yes	String	Common parameter. For more information, please see the list of regions supported by the product.

3. Output Parameters

Parameter Name	Type	Description
RequestId	String	The unique request ID, which is returned for each request. RequestId is required for locating a problem.

4. Example

Example1 Creating tracking set

Input Example

```
POST / HTTP/1.1
Host: cloudaudit.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: CreateAuditTrack
<Common request parameters>

{}
```

Output Example

```
{
"Response": {
"RequestId": "2d4a7fba-bba8-452e-a99e-ccf11fdaa583"
}
}
```

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 NodeJS

- 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
InternalError	Internal error.
InvalidParameter	Parameter error.
InvalidParameterValue.AliasAlreadyExists	The alias already exists.
LimitExceeded.OverAmount	The maximum number of tracking sets has been exceeded.

Data Types

Last updated : 2022-03-16 10:53:26

AttributeKeyDetail

`AttributeKey` value details

Used by actions: GetAttributeKey.

Name	Type	Required	Description
LabelType	String	Yes	Input box type
Starter	String	Yes	Initial display
Order	Integer	Yes	Display sort order
Value	String	Yes	`AttributeKey` value
Label	String	Yes	Tag

AuditSummary

Tracking set overview

Used by actions: ListAudits.

Name	Type	Required	Description
AuditStatus	Integer	No	Tracking set status. 1: enabled, 0: disabled
CosBucketName	String	No	COS bucket name
AuditName	String	No	Tracking set name
LogFilePrefix	String	No	Log prefix

CmqRegionInfo

CMQ region information

Used by actions: ListCmqEnableRegion.

Name	Type	Required	Description
CmqRegionName	String	No	Region description
CmqRegion	String	No	CMQ region

CosRegionInfo

COS region information

Used by actions: ListCosEnableRegion.

Name	Type	Required	Description
CosRegion	String	No	COS region
CosRegionName	String	No	Region description

Event

Log details

Used by actions: DescribeEvents, LookUpEvents.

Name	Type	Required	Description
EventId	String	No	Log ID
Username	String	No	Username
EventTime	String	No	Event Time
CloudAuditEvent	String	No	Log details
ResourceTypeCn	String	No	Description of resource type in Chinese (please use this field as required; if you are using other languages, ignore this field)
ErrorCode	Integer	No	Authentication error code
EventName	String	No	Event name

SecretId	String	No	Certificate ID Note: `null` may be returned for this field, indicating that no valid values can be obtained.
EventSource	String	No	Request source
RequestID	String	No	Request ID
ResourceRegion	String	No	Resource region
AccountID	Integer	No	Root account ID
SourceIPAddress	String	No	Source IP Note: `null` may be returned for this field, indicating that no valid values can be obtained.
EventNameCn	String	No	Description of event name in Chinese (please use this field as required; if you are using other languages, ignore this field)
Resources	Resource	No	Resource pair
EventRegion	String	No	Event region
Location	String	No	IP location

LookupAttribute

Search criterion

Used by actions: DescribeEvents, LookUpEvents.

Name	Type	Required	Description
AttributeKey	String	Yes	Valid values: RequestId, EventName, ReadOnly, Username, ResourceType, ResourceName, AccessKeyId, and EventId Note: `null` may be returned for this field, indicating that no valid values can be obtained.
AttributeValue	String	No	Value of `AttributeValue` Note: `null` may be returned for this field, indicating that no valid values can be obtained.

Resource

Resource type

Used by actions: DescribeEvents, LookUpEvents.

Name	Type	Required	Description
ResourceType	String	No	Resource type
ResourceName	String	No	Resource name Note: `null` may be returned for this field, indicating that no valid values can be obtained.

Error Codes

Last updated : 2022-03-16 10:53:26

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	`Authorization` 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.
InternalError	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
FailedOperation.CreateBucketFail	Failed to create the COS bucket.
InternalError.CmqError	An exception occurred while creating the CMQ queue, probably because the CMQ queue to be created already exists, or

	your account has no permission or has overdue payments.
InternalError.CreateAuditError	An error occurred while creating the tracking set. Submit a ticket for assistance.
InternalError.DeleteAuditError	Failed to delete the tracking set. Submit a ticket for assistance.
InternalError.DescribeAuditError	An error occurred while querying tracking set details. Submit a ticket for assistance.
InternalError.InquireAuditCreditError	An error occurred while querying the number of tracking sets that can be created. Submit a ticket for assistance.
InternalError.ListAuditsError	An internal error occurred while querying the summary of tracking sets. Submit a ticket for assistance.
InternalError.ListCmqEnableRegionError	An internal error occurred. Submit a ticket for assistance.
InternalError.ListCosEnableRegionError	An internal error occurred. Submit a ticket for assistance.
InternalError.SearchError	An internal error occurred. Submit a ticket for assistance.
InternalError.StartLoggingError	An internal error occurred. Submit a ticket for assistance.
InternalError.StopLoggingError	An internal error occurred. Submit a ticket for assistance.
InternalError.UpdateAuditError	An internal error occurred. Submit a ticket for assistance.
InvalidParameter.Time	The parameter must contain the start time and end time and must be an integer timestamp (accurate down to the second).
InvalidParameterValue.AliasAlreadyExists	The alias already exists.
InvalidParameterValue.AuditNameError	The tracking set name is non-compliant.
InvalidParameterValue.CmqRegionError	CloudAudit currently does not support the

	entered CMQ region.
InvalidParameterValue.CosNameError	The entered COS bucket name is non-compliant.
InvalidParameterValue.CosRegionError	CloudAudit currently does not support the entered COS region.
InvalidParameterValue.IsCreateNewBucketError	The value of `IsCreateNewBucket` can be 0 or 1. 0 indicates not to create a bucket, while 1 indicates to create a bucket.
InvalidParameterValue.IsCreateNewQueueError	The value of `IsCreateNewQueue` can be 0 or 1. 0 indicates not to create a queue, while 1 indicates to create a queue.
InvalidParameterValue.IsEnableCmqNotifyError	The value of `IsEnableCmqNotify` can be 0 or 1. 0 indicates not to enable CMQ delivery, while 1 indicates to enable CMQ delivery.
InvalidParameterValue.LogFilePrefixError	The log prefix format is incorrect.
InvalidParameterValue.MaxResult	The maximum number of entries returned in one search is 50.
InvalidParameterValue.QueueNameError	The entered queue name is non-compliant.
InvalidParameterValue.ReadWriteAttributeError	Valid values of the read/write attribute: 1 (read-only), 2 (write-only), 3 (read/write).
InvalidParameterValue.Time	The start time cannot be after the end time.
InvalidParameterValue.attributeKey	Valid values of `AttributeKey` : RequestId, EventName, ReadOnly, Username, ResourceType, ResourceName, AccessKeyId
LimitExceeded.OverAmount	The maximum number of tracking sets has been exceeded.
LimitExceeded.OverTime	Only entries for the last 7 days can be searched for.
MissingParameter.MissAuditName	The tracking set name is missing.

MissingParameter.MissCosBucketName	The COS bucket parameter is missing.
MissingParameter.MissCosRegion	The COS region parameter is missing.
MissingParameter.cmq	If the value of `IsEnableCmqNotify` is 1, `IsCreateNewQueue`, `CmqQueueName`, and `CmqRegion` are required.
ResourceInUse.AlreadyExistsSameAudit	A tracking set with the same name already exists.
ResourceInUse.AlreadyExistsSameAuditCmqConfig	A tracking set with the same CMQ delivery configuration already exists.
ResourceInUse.AlreadyExistsSameAuditCosConfig	A tracking set with the same COS delivery configuration already exists.
ResourceInUse.CosBucketExists	The COS bucket already exists.
ResourceNotFound.AuditNotExist	The tracking set does not exist.