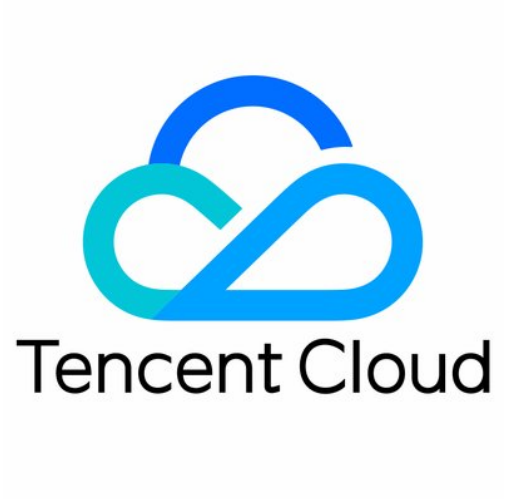
Cloud Log Service

API Document

Product Introduction



Contents

API Document
Request Signature

Last updated：2018-05-18 17:30:02

Preparations

1. Obtain SecretId and SecretKey.
   They are available on the Cloud API Key page of the console.
2. Specify the development language:
   Supported languages include but not limisted to java, php, c sharp, c++, node.js, and python. You need to specify the HMAC-SHA1 function depending on development languages.

Signature Content

The HTTP signature request initiated to CLS through the API is passed by using the standard HTTP Authorization header, as shown in the following example:

```
GET /logset?logset_name=testset HTTP/1.1
Host: ap-shanghai.cls.myqcloud.com
Authorization: q-sign-algorithm=sha1&q-ak=AKIDc9YlmrBcFk4C8sbmXQ8i65XXXXXXXXXX&q-sign-time=1510109254;1510109314&q-key-time=1510109254;1510109314&q-header-list=content-type;host&q-url-param-list=logset_name&q-signature=e8b23b818caf4e33f196f895218bdabdbd1f1423
```

The `Host` field in the request is ${region}.cls.myqcloud.com ("region" is replaced by the actual region of the CLS service), with the following ID:

```
ap-beijing - Beijing
ap-shanghai - Shanghai
ap-guangzhou - Guangzhou
ap-chengdu - Chengdu
```

The signature content in the request is comprised of a number of key-value pairs connected with "&". The format is as follows:

```
q-sign-algorithm=[Algorithm]&q-ak=[SecretId]&q-sign-time=[SignTime]&q-key-time=[KeyTime]&q-header-list=[SignedHeaderList]&q-url-param-list=[SignedParamList]&q-signature=[Signature]
```

Key-Value Description

The key-value pairs (Key=Value) in the signature content are described as follows:

Key	Value	Description
q-sign-algorithm	sha1	Signature algorithm (required). Only sha1 is supported.
q-ak	Parameter [SecretId]	Account's SecretId (required). ID obtained from the console in the preparation above
q-sign-time	Parameter [SignTime]	Signature start and end time (required) which are unix timestamp in seconds and separated by `;`.
q-key-time	Parameter [KeyTime]	It is required, and is the same as q-sign-time.
q-header-list	Parameter [SignedHeaderList]	The key (required) of the HTTP request header that needs adding a signature. The key must be converted to lowercase. Multiple keys must be sorted in lexicographic order and separated by `;`.
q-url-param-list	Parameter [SignedParamList]	The parameter (required) of the Http request Uri that needs adding a signature. The keys must be converted to lowercase. Multiple keys must be sorted in lexicographic order and separated by `;`.
q-signature	Parameter [Signature]	Calculated signature content (required) comprised of lowercase letters

Note: For q-sign-time and q-key-time, the end time must be later than the start time. Otherwise, the signature expires immediately.

Calculation Method

Signature calculation involves two steps:

1. Combine the related information of the Http request to the string HttpRequestInfo, according to a certain format.
2. Use the sha1 algorithm to calculate the hash value for HttpRequestInfo, and then combine the hash value with other specified parameters according to a certain format to generate the original string StringToSign.
3. Encrypt the q-key-time using the SecretKey to get the SignKey.
4. Encrypt the StringToSign using the SignKey to generate the Signature.

Combining HttpRequestInfo

HttpRequestInfo consists of Method, Uri, Headers, and Parameters in the HTTP request, as shown below:

```
HttpRequestInfo = Method + "\n"
+ Uri + "\n"
+ FormatedParameters + "\n"
+ FormatedHeaders + "\n"
```

`\n` indicates a newline escape character, `+` indicates a string concatenation operation. The other parameters are defined as follows:

Field Name	Description
Method	Method (in lowercase letters) used for HTTP requests, such as `get, post`
Uri	The name of resource of the HTTP request, excluding the query string, for example, `/logset`
FormatedParameters	The serialized string of parameters in the Http request's query string, namely the parameters specified in q-url-param-list. If no parameter is specified, an empty string is used. The key and the value are connected with `=`. Different key-value pairs are connected with `&`, and need to be sorted in lexicographical order. The key must be lowercase letters, and the value must be URL encoded.
FormatedHeaders	The header of Http request, namely the Http header specified in q-header-list. If no header is specified, an empty string is used. The key and the value are connected with `=`. Different key-value pairs are connected with `&`, and need to be sorted in lexicographical order, The key must be lowercase letters, and the value must be URL encoded.

Combining StringToSign

StringToSign is composed of the sha1 hash values of q-sign-algorithm, q-sign-time, and HttpRequestInfo, as shown below:

```
StringToSign = q-sign-algorithm + "\n"
+ q-sign-time + "\n"
+ sha1(HttpRequestInfo) + "\n"
```

`\n` indicates a newline escape character. `+` indicates a string concatenation operation. Other parameters have been described above. The sha1 hash of HttpRequestInfo is a lowercase hexadecimal string.

Generating SignKey

Now, the API only supports one digital signature algorithm, hmac-sha1 (default). Its pseudo code is as follows:

```
SignKey = Hexdigest(HMAC-SHA1(q-key-time, SecretKey))
```

`HMAC-SHA1` is the encryption algorithm, and `Hexdigest` is the method used for hexadecimal string conversion. The output results of some languages using the encryption algorithm are hexadecimal strings, so conversion is not needed.

Generating Signature

Now, the API only supports one digital signature algorithm, hmac-sha1 (default). Its pseudo code is as follows:

```
Signature = Hexdigest(HMAC-SHA1(StringToSign, SignKey))
```

`HMAC-SHA1` is the encryption algorithm, and `Hexdigest` is the method used for hexadecimal string conversion. The output results of some languages using the encryption algorithm are hexadecimal strings, so conversion is not needed.

Examples

Take the following SecretId and SecretKey as examples:

```
SecretId = "AKIDc9YlmrBcFk4C8sbmXQ8i65XXXXXXXXXX"
SecretKey = "LUSE4nPK1d4tX5SHyXv6tZXXXXXXXXXX"

StartTime = 1510109254
EndTime = 1510109314
```

Example 1:
To get logset information, the HTTP request is as follows:

```
GET /logset?logset_name=testset HTTP/1.1
Host: ap-shanghai.cls.myqcloud.com
```

For the above request, if a signature is added in the request header Host, the generated string HttpRequestInfo is:

```
get\n/logset\nlogset_name=testset\nhost=ap-shanghai.cls.myqcloud.com\n
```

The original string StringToSign generated using HttpRequestInfo is:

```
sha1\n1510109254;1510109314\n74713a7e01250b81424dac21dced038ee5b8054d\n
```

Encrypt the q-key-time using the SecretKey to get the SignKey:

```
a4501294d3a835f8dab6caf5c19837dd19eef357
```

Encrypt the StringToSign using the SignKey to generate the Signature:

```
42a7a1d1b44f14ae39a5e7fc3172feec6a08b197
```

The combined signature Authorization is:

```
q-sign-algorithm=sha1&q-ak=AKIDc9YlmrBcFk4C8sbmXQ8i65XXXXXXXXXX&q-sign-time=1510109254;1510109314&q-key-time=1510109254;1510109314&q-header-list=host&q-url-param-list=logset_name&q-signature=42a7a1d1b44f14ae39a5e7fc3172feec6a08b197
```

The final request content is:

```
GET /logset?logset_name=testset HTTP/1.1
Host: ap-shanghai.cls.myqcloud.com
Authorization: q-sign-algorithm=sha1&q-ak=AKIDc9YlmrBcFk4C8sbmXQ8i65XXXXXXXXXX&q-sign-time=1510109254;1510109314&q-key-time=1510109254;1510109314&q-header-list=host&q-url-param-list=logset_name&q-signature=42a7a1d1b44f14ae39a5e7fc3172feec6a08b197
```

Example 2:

To modify logset information, the HTTP request is as follows:

```
PUT /logset HTTP/1.1
Host: ap-shanghai.cls.myqcloud.com
Content-Type: application/json
Content-Length: 50

{"logset_id":"xxxx-xx-xx-xx-xxxxxxxx","period":30}
```

The md5 value calculated for the body content is:

```
f9c7fc33c7eab68dfa8a52508d1f4659
```

For the above request, if a signature is added in the request header Host, the generated string HttpRequestInfo is:

```
put\n/logset\n\ncontent-md5=f9c7fc33c7eab68dfa8a52508d1f4659&content-type=application/json&host=ap-shanghai.cls.myqcloud.com\n
```

The original string StringToSign generated using HttpRequestInfo is:

```
sha1\n1510109254;1510109314\n0ca0242c3d50441fda6aa234d31bea7a7a12a1ea\n
```

Encrypt the q-key-time using the SecretKey to get the SignKey:

```
a4501294d3a835f8dab6caf5c19837dd19eef357
```

Encrypt the StringToSign using the SignKey to generate the Signature:

```
85a55e61de42483ba03bffd07a6c01b8d651af51
```

The combined signature Authorization is:

```
q-sign-algorithm=sha1&q-ak=AKIDc9YlmrBcFk4C8sbmXQ8i65XXXXXXXXXX&q-sign-time=1510109254;1510109314&q-key-time=1510109254;1510109314&q-header-list=content-md5;content-type;host&q-url-param-list=&q-signature=85a55e61de42483ba03bffd07a6c01b8d651af51
```

The final request content is:

```
PUT /logset HTTP/1.1
Host: ap-shanghai.cls.myqcloud.com
Content-Type: application/json
Content-MD5: f9c7fc33c7eab68dfa8a52508d1f4659
Content-Length: 50
Authorization: q-sign-algorithm=sha1&q-ak=AKIDc9YlmrBcFk4C8sbmXQ8i65XXXXXXXXXX&q-sign-time=1510109254;1510109314&q-key-time=1510109254;1510109314&q-header-list=content-md5;content-type;host&q-url-param-list=&q-signature=85a55e61de42483ba03bffd07a6c01b8d651af51

{"logset_id":"xxxx-xx-xx-xx-xxxxxxxx","period":30}
```

Public Request Header

Last updated：2018-05-18 17:30:33

Overview

This document describes public request headers to be used when using CLS APIs. The headers described below will not be discussed in subsequent API documents.

List of Public Request Headers

HTTP Header Name	Description
Host	The request host name, which is different for each region. For example, it is ap-shanghai.cls.myqcloud.com for Shanghai.
Authorization	Signature content. For more information on how to compute the signature, please see Request Signature.
Content-Length	Length of the request Body. If there is no Body, this header can be ignored.
Content-Type	Format of the request Body (application/json, application/x-protobuf), which can be selected according to the detailed API documentation. If there is no Body, this header can be ignored.
Content-MD5	MD5 of the request Body. If there is no Body, this header can be ignored. The calculation result supports lowercase letters.
x-cls-compress-type	Compression method (lz4) used by the request Body, which is only used by the log upload API. If there is no compression, this header can be ignored.

Public Response Header

Last updated：2018-05-18 17:34:53

Overview

This document describes public response headers that will appear when you use CLS APIs. The headers described below will not be discussed in subsequent API documents.

List of Response Headers

The response headers are described as follows:

HTTP Header Name	Description
Content-Length	Length of the response body
Content-Type	Format of the response body (application/json)
x-cls-requestid	The unique ID on the server for this request

Error Codes

Last updated：2018-05-18 17:30:50

Overview

This document describes the error codes and corresponding error messages returned when an error occurs with the request. You can identify the problem based on StatusCode and Body of HTTP. The format of Body is as follows:

```
{
"errorcode" : <ErrorCode>,
"errormessage" : <ErrorMessage>
}
```

Error Codes

HTTP Status Code	Error Code	Error Message
400	InvalidAuthorization	Invalid signature string format
400	InvalidCompressType	The specified x-cls-compress-type is not supported
400	InvalidContent	Message body error. Decompression or resolution failed
400	InvalidContentType	The specified Content-Type is not supported
400	InvalidParam	Required parameters are missing or some parameters are invalid
400	MissingAgentIp	x-cls-agent-ip is missing
400	MissingAgentVersion	x-cls-agent-version is missing
400	MissingAuthorization	Authorization is missing
400	MissingContent	Message body is empty
400	MissingContentType	Content-Type is missing

HTTP Status Code	Error Code	Error Message
400	TopicClosed	Collection feature is disabled for the specified topic
401	Unauthorized	Authentication failed, possibly due to ID error, repeated use of signature, or signature computing error
403	LogsetExceed	Number of logsets exceeds the limit of 10
403	LogSizeExceed	Size of the submitted log exceeds the limit of 5 MB
403	NotAllowed	Operation is not allowed
403	TopicExceed	Number of log topics exceeds the limit of 20
404	IndexNotExist	The specified index rule does not exist
404	LogsetNotExist	The specified logset does not exist
404	MachineGroupNotExist	The specified server group does not exist
404	ShipperNotExist	The specified shipping rule does not exist
404	TopicNotExist	The specified log topic does not exist
405	NotSupported	Operation is not supported
409	IndexConflict	The index rule already exists
409	LogsetConflict	The logset already exists
409	MachineGroupConflict	The server group already exists
409	ShipperConflict	The shipping rule already exists
409	TopicConflict	The log topic already exists
500	InternalError	Internal error