API Gateway

Development Guide

Product Documentation

Contents

Development Guide
Serverless Framework
Overview

Last updated：2025-04-07 09:35:49

Overview

Well-received in the industry, Serverless Framework allows you to deploy a complete, available serverless application framework without the need to care about underlying resources. It features resource orchestrating, auto scaling, and event driving and covers the full development lifecycle from coding and debugging to testing and deploying, helping you quickly build serverless applications with the aid of Tencent Cloud resources.

Serverless Framework Components

Serverless Components is a scenario-based solution that supports orchestration and organization of multiple cloud resources for different use cases such as Express framework integration and website deployment. It can greatly simplify the configuration and management of cloud resources while interconnecting Tencent Cloud products such as API Gateway, COS, and CAM, so that you can focus more on your business development. For more information.

Benefits and features

Ease of use

Serverless Components is built around your scenarios (e.g., websites, blogs, payment systems, and image services). It abstracts underlying infrastructure configuration and enables you to implement your business scenarios with simple configurations.

Reusability

A serverless component can be easily created and deployed through a very simple `serverless.yml` file. Plus, its JavaScript library `serverless.js` supports extension and reuse with simple syntax.

Fast deployment

The deployment of most serverless components is about 20 times faster than traditional configuration tools. They allow rapid deployment and remote verification which help effectively reduce the workload of local emulation and debugging.

API Gateway component

The API Gateway component is one of the basic components provided by Tencent Serverless Framework. Through this component, you can create, configure, and manage API gateways with speed and ease.

Practical Tutorial

You can start using Serverless Framework CLI through the following practices:

Item	Description
Deploying Express.js application	Use the `Serverless SCF` component to quickly construct an Express.js project.
Deploying Nuxt.js application	Use the `Serverless Components Nuxt.js` component to quickly deploy an SSR project based on Nuxt.js.

Installing Serverless Framework

Last updated：2023-12-22 10:04:20

This document describes how to quickly install Serverless Framework through binary installation or npm.

Installing

Method 1. Binary installation

If Node.js is not installed in your local environment, you can install it directly in binary mode:

Note:

You are recommended to use Node.js 10.0 or above; otherwise, Component v2 may report errors during deployment.

macOS/Linux

Open the command line and enter the following command:

```
$ curl -o- -L https://slss.io/install | bash
```

If you have already installed a binary version, you can run the following command to upgrade it:

```
$ serverless upgrade
```

Windows

Windows supports installation through Chocolatey. Open the command line and enter the following command:

```
$ choco install serverless
```

If you have already installed a binary version, you can run the following command to upgrade it:

```
$ choco upgrade serverless
```

Method 2. Installation through npm

Note:

You are recommended to use Node.js 10.0 or above; otherwise, Component v2 may report errors during deployment.

Run the following command on the command line:

```
$ npm install -g serverless
```

Note:

If macOS prompts that you have no permission, you need to run `sudo npm install -g serverless` for installation.
If you have already installed Serverless Framework, you can run the following command to upgrade it to the latest version:

```
$ npm update -g serverless
```

If Node.js is not installed in your environment, you can install it as instructed in Node.js Installation Guide based on your system environment.

Viewing Version Information

After the installation is completed, run the `serverless -v` command to view the version information of Serverless Framework:

```
$ serverless -v
```

Creating and Deploying API Gateway Service

Last updated：2023-12-22 10:04:30

Operation Scenarios

This document describes how to use the API Gateway component provided by Serverless Framework to quickly create and deploy an API Gateway service/API.

Prerequisites

You have installed Serverless Framework.

Directions

Configuring API Gateway service

Create the `serverless.yml` file locally:

```
touch serverless.yml
```

Configure `serverless.yml` as follows:

```
# serverless.yml

component: apigateway # Component name, which is required. `apigateway` is used in
name: apigwDemo # Instance name, which is required
org: orgDemo # Organization information, which is optional. The default value is th
app: appDemo # Next.js application name, which is optional
stage: dev # Information for identifying environment, which is optional. The defaul

inputs:
  region: ap-guangzhou
  protocols:
    - http
    - https
  serviceName: serverless
  environment: release
  endpoints:
    - path: /
      protocol: HTTP
```

```
        method: GET
        apiName: index
        function:
          functionName: myFunction
```

Detailed Configuration >>

Deploying API Gateway service

Run the following command to deploy by scanning code:

```
sls deploy
```

Note:

To grant persistent permission, please see Account Configuration.

Removing deployed service

You can run the following command to remove the deployed service:

```
sls remove
```

Account configuration (optional)

Currently, you can scan a QR code to log in to the CLI by default. If you want to configure persistent environment variables/key information, you can also create a local `.env` file:

```
touch .env # Tencent Cloud configuration information
```

Configure Tencent Cloud's `SecretId` and `SecretKey` information in the `.env` file and save it:

```
# .env
TENCENT_SECRET_ID=123
TENCENT_SECRET_KEY=123
```

Note:

If you don't have a Tencent Cloud account yet, please sign up first.

If you already have a Tencent Cloud account, you can get `SecretId` and `SecretKey` in API Key Management.

Importing APIs
Defining APIs

Last updated：2023-12-22 10:04:42

Standard RESTful APIs can be defined using the OpenAPI Specification. Generally, OpenAPI documents have 3 required objects:
openapi: version number of the OpenAPI Specification.
info: API metadata.
paths: API request path and operation.
This document describes the mappings between the OpenAPI Specification and the API Gateway by object.

openapi

The API Gateway supports OpenAPI Specification 3.0.0.

info

The following table describes the mappings between OpenAPI and the info object of the API Gateway.

OpenAPI Object	Data Type	OpenAPI Object Description	API Gateway Object
info.title	String	Name of the service to which an API belongs	Not used
info.description	String	Description of the service to which an API belongs	Not used
info.version	String	Version number	Not used

paths

The following table describes the mappings between OpenAPI and the paths object of the API Gateway.

OpenAPI Object	Data Type	OpenAPI Object Description	API Gateway Object
paths.path	Object	API request path	API frontend request path
operation.operationId	String	API name	API name

operation.description	String	API description	Not used
operation.parameters	Object	API request parameter	API request parameter
operation.responses	String	API response	Not used

Note:

For the object definition in OpenAPI Specification 3.0.0, see OpenAPI Specification.

For the process of importing APIs, see Importing APIs.

For complete examples of importing APIs, see Examples of Importing APIs.

Sample for Importing APIs

Last updated：2023-12-22 10:04:52

This document describes how to import APIs on different backends. The following takes the YAML format as an example.

Mock backend

```
openapi: 3.0.0
info:
  title: test
  version: 1.0.1
paths:
  /:
    get:
      operationId: test
      responses:
        '200':
          description: The list of possible responsesas they are returned from
executing
            this operation.
      x-apigw-api-business-type: NORMAL
      x-apigw-api-type: NORMAL
      x-apigw-backend:
        MockReturnHttpHeaders: []
        MockReturnHttpStatusCode: 200
        ServiceMockReturnMessage: success
        ServiceType: MOCK
      x-apigw-cors: false
      x-apigw-protocol: HTTP
      x-apigw-service-timeout: 15
```

Proxy backend

```
openapi: 3.0.0
info:
  title: testa
  version: 1.0.1
paths:
  /proxy:
    get:
```

```
      operationId: test
      responses:
        '200':
          description: The list of possible responsesas they are returned from
executing
            this operation.
      x-apigw-api-business-type: NORMAL
      x-apigw-api-type: NORMAL
      x-apigw-backend:
        ServiceConfig:
          Method: GET
          Path: /
          Url: http://cloud.tencent.com
        ServiceType: HTTP
      x-apigw-cors: false
      x-apigw-protocol: HTTP
      x-apigw-service-timeout: 15
```

VPC service backend

```
openapi: 3.0.0
info:
  title: test
  version: 1.0.1
paths:
  /:
    get:
      operationId: test
      responses:
        '200':
          description: The list of possible responsesas they are returned from
executing
            this operation.
      x-apigw-api-business-type: NORMAL
      x-apigw-api-type: NORMAL
      x-apigw-backend:
        ServiceConfig:
          Method: GET
          Path: /
          Product: clb
          UniqVpcId: vpc-xxxxxx
          Url: http://172.x.x.x:8xxx
        ServiceType: HTTP
      x-apigw-const-paramters:
```

```
      - DefaultValue: xxx
        Desc: "xxxx backend host"
        Name: Host
        Position: HEADER
      x-apigw-cors: false
      x-apigw-protocol: HTTP
      x-apigw-service-timeout: 15
```

SCF event backend

```
openapi: 3.0.0
info:
  title: testa
  version: 1.0.1
paths:
  /scf:
    get:
      operationId: test
      responses:
        '200':
          description: The list of possible responsesas they are returned from
executing
            this operation.
      x-apigw-api-business-type: NORMAL
      x-apigw-api-type: NORMAL
      x-apigw-backend:
        IsBase64Encoded: false
        ServiceScfFunctionName: APIGWCustomRespDemo-xxxxx
        ServiceScfFunctionNamespace: default
        ServiceScfFunctionQualifier: $DEFAULT
        ServiceScfFunctionType: EVENT
        ServiceScfIsIntegratedResponse: false
        ServiceType: SCF
      x-apigw-cors: false
      x-apigw-protocol: HTTP
      x-apigw-service-timeout: 15
```

SCF HTTP-triggered function backend

```
openapi: 3.0.0
info:
```

```
  title: testa
  version: 1.0.1
paths:
  /scf:
    get:
      operationId: test
      responses:
        '200':
          description: The list of possible responsesas they are returned from
executing
            this operation.
      x-apigw-api-business-type: NORMAL
      x-apigw-api-type: NORMAL
      x-apigw-backend:
        IsBase64Encoded: false
        ServiceScfFunctionName: flask_demo-xxxxxxxxx
        ServiceScfFunctionNamespace: default
        ServiceScfFunctionQualifier: $DEFAULT
        ServiceScfFunctionType: HTTP
        ServiceScfIsIntegratedResponse: false
        ServiceType: SCF
      x-apigw-cors: false
      x-apigw-protocol: HTTP
      x-apigw-service-timeout: 15
```

COS backend

```
openapi: 3.0.0
info:
  title: test
  version: 1.0.1
paths:
  /cos:
    get:
      operationId: test
      responses:
        '200':
          description: The list of possible responsesas they are returned from
executing
            this operation.
      x-apigw-api-business-type: NORMAL
      x-apigw-api-type: NORMAL
      x-apigw-backend:
        ServiceConfig:
```

```
          CosConfig:
            Action: GetObject
            Authorization: true
            BucketName: xxxxxxx
            PathMatchMode: FullPath
          Path: /
        ServiceType: COS
      x-apigw-cors: false
      x-apigw-protocol: HTTP
      x-apigw-service-timeout: 15
```

TSF microservice API backend

```
openapi: 3.0.0
info:
  title: SCF_API_SERVICE
  version: 1.0.1
paths:
  /:
    get:
      operationId: test
      parameters:
      - description: ""
        in: header
        name: X-MicroService-Name
        required: true
        schema:
          type: string
      - description: ""
        in: header
        name: X-NameSpace-Code
        required: true
        schema:
          type: string
      responses:
        '200':
          description: The list of possible responsesas they are returned from
executing
            this operation.
      x-apigw-api-business-type: NORMAL
      x-apigw-api-type: TSF
      x-apigw-backend:
        MicroServices:
        - ClusterId: cluster-xxxxxx
```

```
          MicroServiceName: provider-demo
          NamespaceId: namespace-xxxxx
        ServiceConfig:
          Path: /
        ServiceTsfHealthCheckConf:
          ErrorThresholdPercentage: 50
          IsHealthCheck: true
          RequestVolumeThreshold: 20
          SleepWindowInMilliseconds: 5000
        ServiceTsfLoadBalanceConf:
          IsLoadBalance: true
          Method: RoundRobinRule
          SessionStickRequired: false
          SessionStickTimeout: 0
        ServiceType: TSF
      x-apigw-cors: false
      x-apigw-protocol: HTTP
      x-apigw-service-timeout: 15
```

Importing an API in JSON format

The following takes the mock backend as an example. For other types, see the YAML format.

```
{
 "openapi": "3.0.0",
 "info": {
   "description": "importMockAPI",
   "version": "1.0.0",
   "title": "Mock API"
 },
 "paths": {
   "/mock": {
     "get": {
       "description": "Import Mock API Test",
       "operationId": "importMockAPI",
       "responses": {
         "200": {
           "description": "Import Mock API Test"
         }
       }
     }
   }
 }
}
```

Generating Application Authentication Signature
Python

Last updated：2023-12-22 10:05:38

Overview

This document describes how to use the application-enabled authentication mode for the authentication management of your APIs in Python.

Directions

1. In the API Gateway console, create an API and select the authentication type as **App authentication**. To learn more about the authentication types, find the documentation for different types of backends in Creating API - Overview.
2. Release the service where the API resides to an environment. See Service Release and Deactivation.
3. Create an application on the Application page in the console.
4. Select the created application in the application list, click **Bind API**, select the service and API, and click **Submit** to bind the application to the API.
5. Generate signing information in Python by referring to the Sample Code.

Environmental Dependencies

API Gateway provides sample codes for Python 2.7 and Python 3 with the request body in JSON format and form-data format. Select as needed.

Notes

For more information on operations such as application lifecycle management, authorizing an app to access the API, and binding an app to an API, see Application Management.
For the application signature generation process, see Application Authentication.

Sample Code

Sample code for Python 2.7 with the request body in JSON format

```
# -*- coding: utf-8 -*-

import base64
import datetime
import hashlib
import hmac
import json
import requests
from urlparse import urlparse

# Application's `ApiAppKey`
ApiAppKey = 'Your ApiAppKey'
# Application's `ApiAppSecret`
ApiAppSecret = 'Your ApiAppSecret'

# apigw access address

Url = 'http://service-xxx-xxx.gz.apigw.tencentcs.com/'
HTTPMethod = 'GET'  # method
Accept = 'application/json'
ContentType = 'application/json'

urlInfo = urlparse(Url)
Host = urlInfo.hostname
Path = urlInfo.path

# Environment information is not included in the signature path

if Path.startswith(('/release', '/test', '/prepub')) :
    Path = '/' + Path[1:].split('/',1)[1]
Path = Path if Path else '/'

# Splice query parameters. The query parameters need to be sorted in
lexicographical order.

if urlInfo.query :
    queryStr = urlInfo.query
    splitStr = queryStr.split('&')
    splitStr = sorted(splitStr)
    sortStr = '&'.join(splitStr)
    Path = Path + '?' + sortStr
```

```
ContentMD5 = ''
GMT_FORMAT = '%a, %d %b %Y %H:%M:%S GMT'
xDate = datetime.datetime.utcnow().strftime(GMT_FORMAT)

# Modify body content

if HTTPMethod == 'POST' :
    body = { "arg1": "a", "arg2": "b" }
    body_json = json.dumps(body)
    ContentMD5 = base64.b64encode(hashlib.md5(body_json).hexdigest())

# Obtain the signature string

signing_str = 'x-date: %s\\n%s\\n%s\\n%s\\n%s\\n%s' % (
    xDate, HTTPMethod, Accept, ContentType, ContentMD5, Path)

# Compute the signature

sign = hmac.new(ApiAppSecret, msg=signing_str, digestmod=hashlib.sha1).digest()
sign = base64.b64encode(sign)
auth = "hmac id=\\"" + ApiAppKey + "\\", algorithm=\\"hmac-sha1\\",
headers=\\"x-date\\", signature=\\""
sign = auth + sign + "\\""

# Send the request

headers = {
    'Host': Host,
    'Accept': Accept,
    'Content-Type': ContentType,
    'x-date': xDate,
    'Authorization': sign
}
if(HTTPMethod == 'GET'):
    ret = requests.get(Url, headers=headers)
if(HTTPMethod == 'POST'):
    ret = requests.post(Url, headers=headers, data=body_json)

print(ret.headers)
print(ret.text)
```

Sample code for Python 2.7 with the request body in form-data format

```
# -*- coding: utf-8 -*-
```

```
import base64
import datetime
import hashlib
import hmac
import json
import requests
import urllib
from urlparse import urlparse

# Application's `ApiAppKey`
ApiAppKey = 'Your ApiAppKey'
# Application's `ApiAppSecret`
ApiAppSecret = 'Your ApiAppSecret'

# apigw access address

Url = 'http://service-xxx-xxx.gz.apigw.tencentcs.com/'
HTTPMethod = 'POST'  # method
Accept = 'application/json'
ContentType = 'application/x-www-form-urlencoded'

urlInfo = urlparse(Url)
Host = urlInfo.hostname
Path = urlInfo.path

# Environment information is not included in the signature path

if Path.startswith(('/release', '/test', '/prepub')) :
    Path = '/' + Path[1:].split('/',1)[1]
Path = Path if Path else '/'

ContentMD5 = ''
GMT_FORMAT = '%a, %d %b %Y %H:%M:%S GMT'
xDate = datetime.datetime.utcnow().strftime(GMT_FORMAT)

# Modify body content

body = { "arg1": "a", "arg2": "b" }
argBody = urllib.urlencode(body)

# When signing, the form parameters are spliced with the query parameters. The
parameters need to be sorted in lexicographical order.

if urlInfo.query :
    argBody = argBody + '&' + urlInfo.query
```

```
splitStr = argBody.split('&')
argBody = sorted(splitStr)
sortStr = '&'.join(argBody)
Path = Path + '?' + sortStr

# Obtain the signature string

signing_str = 'x-date: %s\\n%s\\n%s\\n%s\\n%s\\n%s' % (
    xDate, HTTPMethod, Accept, ContentType, ContentMD5, Path)

# Compute the signature

sign = hmac.new(ApiAppSecret, msg=signing_str, digestmod=hashlib.sha1).digest()
sign = base64.b64encode(sign)
auth = "hmac id=\\"" + ApiAppKey + "\\", algorithm=\\"hmac-sha1\\",
headers=\\"x-date\\", signature=\\""
sign = auth + sign + "\\""

# Send the request

headers = {
    'Host': Host,
    'Accept': Accept,
    'Content-Type': ContentType,
    'x-date': xDate,
    'Authorization': sign
}
if(HTTPMethod == 'GET'):
    ret = requests.get(Url, headers=headers)
if(HTTPMethod == 'POST'):
    ret = requests.post(Url, headers=headers, data=body)

print(ret.headers)
print(ret.text)
```

Sample code for Python 2.7 with the request body in multipart/form-data format

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

import base64
import datetime
import hashlib
import hmac
import httplib, mimetypes
from urlparse import urlparse
```

```
# Application's `ApiAppKey`
ApiAppKey = 'Your ApiAppKey'
# Application's `ApiAppSecret`
ApiAppSecret = 'Your ApiAppSecret'
Url = 'http://service-xxx-xxx.gz.apigw.tencentcs.com/'

# Set form parameters
# fields is a sequence of (name, value) elements for regular form fields.
# files is a sequence of (name, filename, value) elements for data to be
uploaded as files
#Example:
#Fields = [("arg1", "a"), ("arg2", "b")]
#Files = [("file", "@test.txt", open("test.txt", "r").read())]
Fields = []
Files = []

HTTPMethod = 'POST'
Accept = 'application/json'

urlInfo = urlparse(Url)
Host = urlInfo.hostname
Path = urlInfo.path

# Environment information is not included in the signature path
if Path.startswith(('/release', '/test', '/prepub')) :
    Path = '/' + Path[1:].split('/',1)[1]
Path = Path if Path else '/'

# Splice query parameters. The query parameters need to be sorted in
lexicographical order.
if urlInfo.query :
    queryStr = urlInfo.query
    splitStr = queryStr.split('&')
    splitStr = sorted(splitStr)
    sortStr = '&'.join(splitStr)
    Path = Path + '?' + sortStr

GMT_FORMAT = '%a, %d %b %Y %H:%M:%S GMT'
xDate = datetime.datetime.utcnow().strftime(GMT_FORMAT)

def post_multipart(host, selector, fields, files):
    content_type, body = encode_multipart_formdata(fields, files)

    ContentMD5 = base64.b64encode(hashlib.md5(body).hexdigest())
    # Obtain the signature string
    signing_str = 'x-date: %s\\n%s\\n%s\\n%s\\n%s\\n%s' % (
```

```
        xDate, HTTPMethod, Accept, content_type, ContentMD5, Path)

    sign = hmac.new(ApiAppSecret, msg=signing_str,
digestmod=hashlib.sha1).digest()
    sign = base64.b64encode(sign)
    auth = "hmac id=\\"" + ApiAppKey + "\\", algorithm=\\"hmac-sha1\\",
headers=\\"x-date\\", signature=\\""
    sign = auth + sign + "\\""

    h = httplib.HTTPConnection(host)
    h.putrequest(HTTPMethod, selector)
    h.putheader('content-type', content_type)
    h.putheader('content-length', str(len(body)))
    h.putheader('accept', Accept)
    h.putheader('x-date', xDate)
    h.putheader('Authorization', sign)
    h.endheaders()
    h.send(body)
    response = h.getresponse()
    output = response.read()
    return output

def encode_multipart_formdata(fields, files):
    BOUNDARY = '----------ThIs_Is_tHe_bouNdaRY_$'
    CRLF = '\\r\\n'
    L = []
    for (key, value) in fields:
        L.append('--' + BOUNDARY)
        L.append('Content-Disposition: form-data; name="%s"' % key)
        L.append('')
        L.append(value)
    for (key, filename, value) in files:
        L.append('--' + BOUNDARY)
        L.append('Content-Disposition: form-data; name="%s"; filename="%s"' %
(key, filename))
        L.append('Content-Type: %s' % get_content_type(filename))
        L.append('')
        L.append(value)
    L.append('--' + BOUNDARY + '--')
    L.append('')
    body = CRLF.join(L)
    content_type = 'multipart/form-data; boundary=%s' % BOUNDARY
    return content_type, body

def get_content_type(filename):
    return mimetypes.guess_type(filename)[0] or 'application/octet-stream'
```

```
output = post_multipart(Host, Path, Fields, Files)

print(output)
```

Sample code for Python 3 with the request body in form-data format

```
# -*- coding: utf-8 -*-

import base64
import datetime
import hashlib
import hmac
import json
import requests
from urllib.parse import urlparse
from urllib.parse import urlencode

# Application's `ApiAppKey`
ApiAppKey = 'Your ApiAppKey'
# Application's `ApiAppSecret`
ApiAppSecret = 'Your ApiAppSecret'

# apigw access address

Url = 'http://service-xxx-xxx.gz.apigw.tencentcs.com/'
HTTPMethod = 'POST'  # method
Accept = 'application/json'
ContentType = 'application/x-www-form-urlencoded'

urlInfo = urlparse(Url)
Host = urlInfo.hostname
Path = urlInfo.path

# Environment information is not included in the signature path

if Path.startswith(('/release', '/test', '/prepub')) :
    Path = '/' + Path[1:].split('/',1)[1]
Path = Path if Path else '/'

ContentMD5 = ''
GMT_FORMAT = '%a, %d %b %Y %H:%M:%S GMT'
xDate = datetime.datetime.utcnow().strftime(GMT_FORMAT)

# Modify body content
```

```
body = { "arg1": "a", "arg2": "b" }
argBody = urlencode(body)

# When signing, the form parameters are spliced with the query parameters. The
parameters need to be sorted in lexicographical order.

if urlInfo.query :
    argBody = argBody + '&' + urlInfo.query

splitStr = argBody.split('&')
argBody = sorted(splitStr)
sortStr = '&'.join(argBody)
Path = Path + '?' + sortStr

# Obtain the signature string

signing_str = 'x-date: %s\\n%s\\n%s\\n%s\\n%s\\n%s' % (
    xDate, HTTPMethod, Accept, ContentType, ContentMD5, Path)

# Compute the signature

sign = hmac.new(ApiAppSecret.encode(), msg=signing_str.encode(),
digestmod=hashlib.sha1).digest()
sign = base64.b64encode(sign).decode()
auth = "hmac id=\\"" + ApiAppKey + "\\", algorithm=\\"hmac-sha1\\",
headers=\\"x-date\\", signature=\\""
sign = auth + sign + "\\""


# Send the request

headers = {
    'Host': Host,
    'Accept': Accept,
    'Content-Type': ContentType,
    'x-date': xDate,
    'Authorization': sign
}

if HTTPMethod == 'GET' :
    ret = requests.get(Url, headers=headers)
if HTTPMethod == 'POST' :
    ret = requests.post(Url, headers=headers, data=body)

print(ret.headers)
print(ret.text)
```

Sample code for Python 3 with the request body in JSON format

```
# -*- coding: utf-8 -*-
import base64
import datetime
import hashlib
import hmac
import json
import requests
from urllib.parse import urlparse

# Application's `ApiAppKey`
ApiAppKey = 'Your ApiAppKey'
# Application's `ApiAppSecret`
ApiAppSecret = 'Your ApiAppSecret'

# apigw access address
Url = 'http://service-xxx-xxx.gz.apigw.tencentcs.com/'
HTTPMethod = 'GET'  # method
Accept = 'application/json'
ContentType = 'application/json'

urlInfo = urlparse(Url)
Host = urlInfo.hostname
Path = urlInfo.path

# Environment information is not included in the signature path
if Path.startswith(('/release', '/test', '/prepub')) :
    Path = '/' + Path[1:].split('/',1)[1]
Path = Path if Path else '/'

# Splice query parameters. The query parameters need to be sorted in
lexicographical order.
if urlInfo.query :
    queryStr = urlInfo.query
    splitStr = queryStr.split('&')
    splitStr = sorted(splitStr)
    sortStr = '&'.join(splitStr)
    Path = Path + '?' + sortStr

ContentMD5 = ''
GMT_FORMAT = '%a, %d %b %Y %H:%M:%S GMT'
xDate = datetime.datetime.utcnow().strftime(GMT_FORMAT)

# Modify body content
```

```
if HTTPMethod == 'POST' :
    body = { "arg1": "a", "arg2": "b" }
    body_json = json.dumps(body)
    body_md5 = hashlib.md5(body_json.encode()).hexdigest()
    ContentMD5 = base64.b64encode(body_md5.encode()).decode()

# Obtain the signature string
signing_str = 'x-date: %s\\n%s\\n%s\\n%s\\n%s\\n%s' % (
    xDate, HTTPMethod, Accept, ContentType, ContentMD5, Path)

# Compute the signature
sign = hmac.new(ApiAppSecret.encode(), msg=signing_str.encode(),
digestmod=hashlib.sha1).digest()
sign = base64.b64encode(sign).decode()
auth = "hmac id=\\"" + ApiAppKey + "\\", algorithm=\\"hmac-sha1\\",
headers=\\"x-date\\", signature=\\""
sign = auth + sign + "\\""


# Send the request
headers = {
    'Host': Host,
    'Accept': Accept,
    'Content-Type': ContentType,
    'x-date': xDate,
    'Authorization': sign
}

if HTTPMethod == 'GET' :
    ret = requests.get(Url, headers=headers)
if HTTPMethod == 'POST' :
    ret = requests.post(Url, headers=headers, data=body_json)

print(ret.headers)
print(ret.text)
```

Sample code for Python 3 with the request body in multipart/form-data format

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

import base64
import datetime
import hashlib
import hmac
import http.client as httplib
```

```
import mimetypes
from urllib.parse import urlparse


# Application's `ApiAppKey`
ApiAppKey = 'Your ApiAppKey'
# Application's `ApiAppSecret`
ApiAppSecret = 'Your ApiAppSecret'
Url = 'http://service-xxx-xxx.gz.apigw.tencentcs.com/'

# Set form parameters
# fields is a sequence of (name, value) elements for regular form fields.
# files is a sequence of (name, filename, value) elements for data to be
uploaded as files
#Example:
#Fields = [("arg1", "a"), ("arg2", "b")]
#Files = [("file", "@test.txt", open("test.txt", "r").read())]
Fields = []
Files = []

HTTPMethod = 'POST'
Accept = 'application/json'

urlInfo = urlparse(Url)
Host = urlInfo.hostname
Path = urlInfo.path

# Environment information is not included in the signature path
if Path.startswith(('/release', '/test', '/prepub')) :
    Path = '/' + Path[1:].split('/',1)[1]
Path = Path if Path else '/'

# Splice query parameters. The query parameters need to be sorted in
lexicographical order.
if urlInfo.query :
    queryStr = urlInfo.query
    splitStr = queryStr.split('&')
    splitStr = sorted(splitStr)
    sortStr = '&'.join(splitStr)
    Path = Path + '?' + sortStr

GMT_FORMAT = '%a, %d %b %Y %H:%M:%S GMT'
xDate = datetime.datetime.utcnow().strftime(GMT_FORMAT)

def post_multipart(host, selector, fields, files):
    content_type, body = encode_multipart_formdata(fields, files)
```

```
    body_md5 = hashlib.md5(body.encode()).hexdigest()
    ContentMD5 = base64.b64encode(body_md5.encode()).decode()
    #ContentMD5 =
base64.b64encode((hashlib.md5(body.encode).hexdigest()).encode).decode()
    # Obtain the signature string
    signing_str = 'x-date: %s\\n%s\\n%s\\n%s\\n%s\\n%s' % (
        xDate, HTTPMethod, Accept, content_type, ContentMD5, Path)

    sign = hmac.new(ApiAppSecret.encode(), msg=signing_str.encode(),
digestmod=hashlib.sha1).digest()
    sign = base64.b64encode(sign).decode()
    auth = "hmac id=\\"" + ApiAppKey + "\\", algorithm=\\"hmac-sha1\\",
headers=\\"x-date\\", signature=\\""
    sign = auth + sign + "\\""

    h = httplib.HTTPConnection(host)
    h.putrequest(HTTPMethod, selector)
    h.putheader('content-type', content_type)
    h.putheader('content-length', str(len(body)))
    h.putheader('accept', Accept)
    h.putheader('x-date', xDate)
    h.putheader('Authorization', sign)
    h.endheaders()
    h.send(body.encode())
    response = h.getresponse()
    output = response.read().decode()
    return output

def encode_multipart_formdata(fields, files):
    BOUNDARY = '----------ThIs_Is_tHe_bouNdaRY_$'
    CRLF = '\\r\\n'
    L = []
    for (key, value) in fields:
        L.append('--' + BOUNDARY)
        L.append('Content-Disposition: form-data; name="%s"' % key)
        L.append('')
        L.append(value)
    for (key, filename, value) in files:
        L.append('--' + BOUNDARY)
        L.append('Content-Disposition: form-data; name="%s"; filename="%s"' %
(key, filename))
        L.append('Content-Type: %s' % get_content_type(filename))
        L.append('')
        L.append(value)
    L.append('--' + BOUNDARY + '--')
    L.append('')
    body = CRLF.join(L)
```

```
    content_type = 'multipart/form-data; boundary=%s' % BOUNDARY
    return content_type, body

def get_content_type(filename):
    return mimetypes.guess_type(filename)[0] or 'application/octet-stream'

output = post_multipart(Host, Path, Fields, Files)

print(output)
```

JavaScript

Last updated：2023-12-22 10:05:50

Scenario

This document describes how to use the application-enabled authentication mode for the authentication management of your APIs in JavaScript.

Operation Directions

1. In the API Gateway console, create an API and select the authentication type as **App authentication**. To learn more about the authentication types, please find the documentation for different types of backends in Creating API - Overview.
2. Release the service where the API resides to an environment. See Service Release and Deactivation.
3. Create an application on the Application page in the console.
4. Select the created application in the application list, click **Bind API**, select the service and API, and click **Submit** to bind the application to the API.
5. Generate signing information in JavaScript by referring to the Sample Code.

Environmental Dependencies

API Gateway provides sample codes with the request body in JSON format and form-data format. Please select as needed.

Note

For more information on operations such as application lifecycle management, authorizing an app to access the API, and binding an app with an API, please see Application Management.
For the application signature generation process, please see Application Authentication.

Sample Code

Sample codes with the request body in JSON format

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

// Application's `ApiAppKey`
const apiAppKey = 'APIDLIA6tMfqsinsadaaaaaaaapHLkQ1z0kO5n5P'
// Application ApiAppSecret
const apiAppSecret = 'Dc44ACV2Da3Gm9JVaaaaaaaaumYRI4CZfVG8Qiuv'

const dateTime = new Date().toUTCString()
const body = {
  arg1: 'a',
  arg2: 'b',
}
const md5 = crypto.createHash('md5').update(JSON.stringify(body),
'utf8').digest('hex')
const contentMD5 = Buffer.from(md5).toString('base64')
const options = {
  hostname: 'service-xxxxxxxx-1234567890.gz.apigw.tencentcs.com',
  port: 443,
  path: '/data',
  method: 'POST',
  headers: {
    Accept: 'application/json',
    'Content-Type': 'application/json',
    'Content-MD5': contentMD5,
    'Content-Length': JSON.stringify(body).length,
    'x-date': dateTime,
  },
}

const signingStr = [
  `x-date: ${dateTime}`,
  options.method,
  options.headers.Accept,
  options.headers['Content-Type'],
  contentMD5,
  options.path,
].join('\\n')
const signing = crypto.createHmac('sha1', apiAppSecret).update(signingStr,
'utf8').digest('base64')
const sign = `hmac id="${apiAppKey}", algorithm="hmac-sha1", headers="x-date",
signature="${signing}"`
options.headers.Authorization = sign

const req = https.request(options, (res) => {
  console.log(`STATUS: ${res.statusCode}`)
```

```
  res.on('data', (chunk) => {
    console.log('BODY: ' + chunk)
  })
})
req.on('error', (error) => {
  console.error(error)
})
req.write(JSON.stringify(body))
req.end()
```

Sample code with the request body in form-data format

```
const https = require('https')
const crypto = require('crypto')
const querystring = require('querystring')
const url = require('url')

// Application's `ApiAppKey`
const apiAppKey = 'APIDLIA6tMfqsinsadaaaaaaaapHLkQ1z0kO5n5P'
// Application ApiAppSecret
const apiAppSecret = 'Dc44ACV2Da3Gm9JVaaaaaaaaumYRI4CZfVG8Qiuv'

const dateTime = new Date().toUTCString()
const body = {
  arg1: 'a',
  arg2: 'b',
}
const contentMD5 = ''
const options = {
  hostname: 'service-xxxxxxxx-1234567890.gz.apigw.tencentcs.com',
  port: 443,
  path: '/data',
  method: 'POST',
  headers: {
    Accept: 'application/json',
    'Content-Type': 'application/x-www-form-urlencoded',
    'x-date': dateTime,
  },
}

// Splicing form parameter and query parameter and sort them according to the
dictionary
const parsedPath = url.parse(options.path, true)
const sortedQueryParams = sortQueryParams({ ...body, ...parsedPath.query })
const signingStr = buildSignStr(sortedQueryParams)
```

```
const signing = crypto.createHmac('sha1', apiAppSecret).update(signingStr,
'utf8').digest('base64')
const sign = `hmac id="${apiAppKey}", algorithm="hmac-sha1", headers="x-date",
signature="${signing}"`

options.headers.Authorization = sign

// Send the request
const req = https.request(options, (res) => {
  console.log(`STATUS: ${res.statusCode}`)
  res.on('data', (chunk) => {
    console.log('BODY: ' + chunk)
  })
})
req.on('error', (error) => {
  console.error(error)
})
req.write(querystring.stringify(body))
req.end()

function sortQueryParams(body) {
  const keys = Object.keys(body).sort()
  let signKeys = []
  for (let i = 0; i < keys.length; i++) {
    signKeys.push(keys[i])
  }
  // Sort in lexicographical order
  return signKeys.sort()
}

function buildSignStr(sorted_body) {
  const keyStr = sorted_body
    .map((item) => {
      return `${item}=${body[item]}`
    })
    .join('&')
  return [
    `x-date: ${dateTime}`,
    options.method,
    options.headers.Accept,
    options.headers['Content-Type'],
    contentMD5,
    `${parsedPath.pathname}${keyStr ? `?${keyStr}` : ''}`,
  ].join('\\n')
}
```

PHP

Last updated：2023-12-22 10:06:06

Overview

This document describes how to manage access to your APIs through application authentication in PHP.

Directions

1. In the API Gateway console, create an API and select the authentication type as **App authentication**. To learn more about the authentication types, find the documentation for different types of backends in Creating API - Overview.
2. Release the service where the API resides to an environment. See Service Release and Deactivation.
3. Create an application on the Application page in the console.
4. Select the created application in the application list, click **Bind API**, select the service and API, and click **Submit** to bind the application to the API.
5. Generate signing information in PHP by referring to the Sample Code.

Environmental Dependencies

This code sample is for PHP 7, and you may need to make changes for other PHP versions as appropriate.
API Gateway provides sample codes with the request body in JSON format and form-data format. Select as needed.

Notes

For more information on operations such as application lifecycle management, authorizing an app to access the API, and binding an app to an API, see Application Management.
For the application signature generation process, see Application Authentication.

Sample Code

Basic code

```
<?php
```

```
const FORM_URLENCODED = "application/x-www-form-urlencoded";

/**
 * Generate a sorted query parameter string from parameter array.
 * such as array('b' => 1, 'a' => 2) to "a=1&b=2"
 *
 * @param array $paramArr
 * @return string sorted query string
 */
function getSortedParameterStr($paramArr)
{
    ksort($paramArr);
    $tmpArr = array();
    foreach ($paramArr as $k => $v) {
        $tmp = $k;
        if (!empty($v)) {
            $tmp .= ("=" . $v);
        }
        array_push($tmpArr, $tmp);
    }

    return join('&', $tmpArr);
}

/**
 * Send a HTTP request with App Authorization
 *
 * @param $apiAppKey string API App Key
 * @param $apiAppSecret string API App Secret
 * @param $method string HTTP Method of API
 * @param $url string Request URL of API, note that environment path (/release)
is not allowed
 * @param $contentType string Request Content-Type header, set empty if request
body is not needed
 * @param $acceptHeader string Accept HTTP request header
 * @param $reqBody string Request Body, set null if request body is not needed
 * @param $formParam array form parameters array, set null if not form request
 * @param $algorithm string Encryption algorithm: sha1, sha256, sha384, sha512,
SM3, default to sha1
 * @param $customHeaders array Custom HTTP Headers, such as `array('x-header-a'
=> 1)`
 */
function sendRequestWithAppAuth($apiAppKey, $apiAppSecret, $method, $url,
$contentType, $acceptHeader,
                                $reqBody=null, $formParam=null,
$algorithm=null, $customHeaders=null)
```

```
{
    $contentMD5 = "";
    $isForm = ($contentType == FORM_URLENCODED);
    // Note: ContentMd5 is empty for application/x-www-form-urlencoded request
    if ($isForm) {
        assert(!is_null($formParam), "formParam is required for form request");
        // generate request body from form parameters
        $reqBody = getSortedParameterStr($formParam);
    } elseif (!is_null($reqBody)) {
        // get content md5 for signing the request later
        $contentMD5 = base64_encode(md5($reqBody));
    }

    if (null === $algorithm) {
        $algorithm = "sha1";
    }

    // ====================================
    // STEP 1: Generate the string to sign
    // ====================================

    echo "1. URL:\\n $url\\n";

    // Note:
    // 1. parameters needs to be sorted in alphabetical order
    // 2. parameters include both query parameters and form parameters
    $paramArr = array();
    $parsedUrl = parse_url($url);
    if (!is_null($parsedUrl['query']) && !empty($parsedUrl['query'])) {
        parse_str($parsedUrl['query'], $paramArr);
    }
    if (!empty($formParam)) {
        $paramArr = array_merge($paramArr, $formParam);
    }

    $pathAndParam = $parsedUrl['path'];
    if (!empty($paramArr)){
        $pathAndParam = $pathAndParam . '?' . getSortedParameterStr($paramArr);
    }

    $xDateHeader = gmstrftime('%a, %d %b %Y %T %Z', time());
    $strToSign = sprintf("x-date: %s\\n%s\\n%s\\n%s\\n%s\\n%s",
        $xDateHeader, $method, $acceptHeader, $contentType, $contentMD5,
$pathAndParam);

    // Print stringToSign for debugging if authorization failed with 401
    $strToSignDebug = str_replace("\\n", "#", $strToSign);
```

```
    echo "2. StringToSign:\\n $strToSignDebug\\n";

    //
================================================================================
    // STEP 2: Generate the signature (Authorization header) based on the
stringToSign
    //
================================================================================

    // Encode the string with HMAC and base64.
    $sign = base64_encode(hash_hmac($algorithm, $strToSign, $apiAppSecret,
TRUE));
    $authHeader = sprintf(
        'hmac id="%s", algorithm="hmac-%s", headers="x-date", signature="%s"',
        $apiAppKey, $algorithm, $sign
    );

    $headers = array(
        'Host:' . $parsedUrl['host'],
        'Accept:' . $acceptHeader,
        'X-Date:' . $xDateHeader,
        'Authorization:' . $authHeader,
    );
    if (!empty($contentType)) {
        array_push($headers, "Content-Type:" . $contentType);
    }
    if (!empty($contentMD5)) {
        array_push($headers, "Content-MD5:" . $contentMD5);
    }
    if (!is_null($customHeaders) && is_array($customHeaders)) {
        foreach ($customHeaders as $k => $v) {
            array_push($headers, $k . ":" . $v);
        }
    }

    echo "3. Request Headers:\\n";
    var_dump($headers);

    // ============================
    // STEP 3: Send the API request
    // ============================

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($ch, CURLOPT_TIMEOUT, 60);
    if (!empty($reqBody)) {
```

```
        curl_setopt($ch, CURLOPT_POSTFIELDS, $reqBody); // only required if
request body is present
    }
    curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, $method);
    $data = curl_exec($ch);
    if (curl_errno($ch)) {
        print "Error: " . curl_error($ch);
    } else {
        echo "Response: \\n";
        var_dump($data);
        curl_close($ch);
    }
}
```

GET sample request

```
// ============================================================
// Example 1: Send a GET request without a request body
// ============================================================
$apiAppKey = '<your_app_key>';
$apiAppSecret = '<your_app_secret>';

$httpMethod = "GET";
$acceptHeader = "application/json";

// Note: environment path, such as `/release`, is not supported with App
Authorization
$url = 'https://service-xxxx-xxxx.gz.apigw.tencentcs.com/testmock?b=1&a=2';
sendRequestWithAppAuth($apiAppKey, $apiAppSecret, $httpMethod, $url, "",
$acceptHeader);
```

POST JSON sample request

```
// ============================================================
// Example 2: Send a POST request with JSON request body
// ============================================================
$apiAppKey = '<your_app_key>';
$apiAppSecret = '<your_app_secret>';

$httpMethod = "POST";
$acceptHeader = "application/json";
$contentType = "application/json";
```

```
// Note: environment path, such as `/release`, is not supported with App
Authorization
$url = 'https://service-xxxx-xxxx.gz.apigw.tencentcs.com/testmock?b=1&a=2';
$jsonBody = "{\\"data\\":1}";
sendRequestWithAppAuth($apiAppKey, $apiAppSecret, $httpMethod, $url,
    $contentType, $acceptHeader, $jsonBody);
```

POST form-urlencoded sample request

```
$apiAppKey = '<your_app_key>';
$apiAppSecret = '<your_app_secret>';

$httpMethod = "POST";
$acceptHeader = "application/json";
$contentType = "application/x-www-form-urlencoded";

// Note: environment path, such as `/release`, is not supported with App
Authorization
$url = 'https://service-xxxx-xxxx.gz.apigw.tencentcs.com/testmock?b=1&a=2';
$customHeaders = array("x-custom-header" => 1);
$formParam = array('id' => 1, 'name' => 'tencent');
sendRequestWithAppAuth($apiAppKey, $apiAppSecret, $httpMethod, $url,
    $contentType, $acceptHeader, null, $formParam, null, $customHeaders);
```

Go

Last updated：2023-12-22 10:06:16

Overview

This document describes how to manage access to your APIs through application authentication in Go.

Directions

1. In the API Gateway console, create an API and select the authentication type as **App authentication**. To learn more about the authentication types, please find the documentation for different types of backends in Creating API - Overview.
2. Release the service where the API resides to an environment. See Service Release and Deactivation.
3. Create an application on the Application page in the console.
4. Select the created application in the application list, click **Bind API**, select the service and API, and click **Submit** to bind the application to the API.
5. Generate signing information in Go by referring to the Sample Code.

Environmental Dependencies

API Gateway provides sample codes with the request body in JSON format and form-data format. Please select as needed.

Notes

For more information on operations such as application lifecycle management, authorizing an app to access the API, and binding an app to an API, please see Application Management.
For the application signature generation process, please see Application Authentication.

Sample Code

Sample code with the request body in JSON format

```
package main

import (
    "crypto/hmac"
    "crypto/md5"
    "crypto/sha1"
    "encoding/base64"
    "encoding/hex"
    "fmt"
    "io/ioutil"
    "log"
    "net/http"
    "net/url"
    "sort"
    "strings"
    "time"
)

func main() {
    // Application's `ApiAppKey`
    const ApiAppKey = "Your ApiAppKey"
    // Application's `ApiAppSecret`
    const ApiAppSecret = "Your ApiAppSecret"

    const Url = "http://service-xxx-xxx.gz.apigw.tencentcs.com/"

    const GmtFormat = "Mon, 02 Jan 2006 15:04:05 GMT"
    const HTTPMethod = "GET"
    const Accept = "application/json"
    const ContentType = "application/json"

    // Parse `Host` and `Path` according to `Url`
    u, err := url.Parse(Url)
    if err != nil {
        log.Fatal(err)
    }
    Host:= u.Hostname()
    Path:= u.Path
    Query := u.RawQuery

    // Environment information is not included in the signature path
    if strings.HasPrefix(Path, "/release") {
        Path = strings.TrimPrefix(Path, "/release")
    }else if strings.HasPrefix(Path, "/test") {
        Path = strings.TrimPrefix(Path, "/test")
    }else if strings.HasPrefix(Path, "/prepub") {
```

```
        Path = strings.TrimPrefix(Path, "/prepub")
    }

    if Path == "" {
        Path = "/"
    }

    // Splice query parameters. The query parameters need to be sorted in
lexicographical order.
    if len(Query) > 0 {
        args, _ := url.ParseQuery(Query)

        var keys []string
        for k := range args {
            keys = append(keys, k)
        }
        sort.Strings(keys)

        sortQuery := ""
        for _, k := range keys {
            if args[k][0] != "" {
                sortQuery = sortQuery + "&" + k + "=" + args[k][0]
            } else {
                sortQuery = sortQuery + "&" + k
            }
        }
        sortQuery = strings.TrimPrefix(sortQuery, "&")

        Path = Path + "?" + sortQuery
    }

    // Get the current UTC time
    xDate := time.Now().UTC().Format(GmtFormat)
    ContentMD5 := ""
    bodyStr := `{"arg1":"a","arg2":"b"}`
    if HTTPMethod == "POST" {
        h := md5.New()
        h.Write([]byte(bodyStr))
        md5Str := hex.EncodeToString(h.Sum(nil))
        ContentMD5 = base64.StdEncoding.EncodeToString([]byte(md5Str))
    }

    // Construct the signature
    signingStr := fmt.Sprintf("x-date: %s\\n%s\\n%s\\n%s\\n%s\\n%s", xDate,
HTTPMethod, Accept, ContentType,
        ContentMD5, Path)
    mac := hmac.New(sha1.New, []byte(ApiAppSecret))
```

```
    _, err= mac.Write([]byte(signingStr))
    if err != nil {
        log.Fatal(err)
    }
    signature := base64.StdEncoding.EncodeToString(mac.Sum(nil))
    sign := fmt.Sprintf("hmac id=\\"%s\\", algorithm=\\"hmac-sha1\\",
headers=\\"x-date\\", signature=\\"%s\\"",
        ApiAppKey, signature)

    // Construct the request
    headers := map[string]string{
        "Host":          Host,
        "Accept":        Accept,
        "Content-Type":  ContentType,
        "x-date":        xDate,
        "Authorization": sign,
    }

    // Send the request
    req, err := http.NewRequest(HTTPMethod, Url, strings.NewReader(bodyStr))
    if err != nil {
        log.Fatal(err)
    }

    for k, v := range headers {
        req.Header.Add(k, v)
    }

    res, err := http.DefaultClient.Do(req)
    if err != nil {
        log.Fatal(err)
    }
    defer res.Body.Close()

    resBody, _ := ioutil.ReadAll(res.Body)

    fmt.Println(string(resBody))
}
```

Sample code with the request body in form-data format

```
package main

import (
```

```
    "crypto/hmac"
    "crypto/sha1"
    "encoding/base64"
    "fmt"
    "io/ioutil"
    "log"
    "net/http"
    "net/url"
    "sort"
    "strings"
    "time"
)

func main() {
    // Application's `ApiAppKey`
    const ApiAppKey = "Your ApiAppKey"
    // Application's `ApiAppSecret`
    const ApiAppSecret = "Your ApiAppSecret"

    const Url = "http://service-xxx-xxx.gz.apigw.tencentcs.com/"

    const GmtFormat = "Mon, 02 Jan 2006 15:04:05 GMT"
    const HTTPMethod = "POST"
    const Accept = "application/json"
    const ContentType = "application/x-www-form-urlencoded"

    // Parse `Host` and `Path` according to `Url`
    u, err := url.Parse(Url)
    if err != nil {
        log.Fatal(err)
    }
    Host := u.Hostname()
    Path := u.Path
    Query := u.RawQuery

    // Environment information is not included in the signature path
    if strings.HasPrefix(Path, "/release") {
        Path = strings.TrimPrefix(Path, "/release")
    }else if strings.HasPrefix(Path, "/test") {
        Path = strings.TrimPrefix(Path, "/test")
    }else if strings.HasPrefix(Path, "/prepub") {
        Path = strings.TrimPrefix(Path, "/prepub")
    }

    if Path == "" {
        Path = "/"
    }
```

```
    // Splice query parameters. The query parameters need to be sorted in
lexicographical order. In this demo, it is assumed the parameters are already
sorted.
    if len(Query) > 0 {
        Path = Path + "?" + Query
    }

    // Get the current UTC time
    xDate := time.Now().UTC().Format(GmtFormat)
    ContentMD5 := ""

    // Request body form data
    body := map[string]string{
        "arg1": "a",
        "arg2": "b",
    }
    var bodyKeys []string
    for k := range body {
        bodyKeys = append(bodyKeys, k)
    }

    var bodyBuilder strings.Builder
    sort.Strings(bodyKeys)
    for _, k := range bodyKeys {
        bodyBuilder.WriteString(fmt.Sprintf("%s=%s&", k, body[k]))
    }
    bodyStr := bodyBuilder.String()
    // Remove the last `&`
    bodyStr = bodyStr[:len(bodyStr) - 1]

    // Construct the signature
    signingStr := fmt.Sprintf("x-date: %s\\n%s\\n%s\\n%s\\n%s\\n%s?%s", xDate,
HTTPMethod, Accept, ContentType,
        ContentMD5, Path, bodyStr)
    mac := hmac.New(sha1.New, []byte(ApiAppSecret))

    _, err      = mac.Write([]byte(signingStr))
    if err != nil {
        log.Fatal(err)
    }
    signature := base64.StdEncoding.EncodeToString(mac.Sum(nil))
    sign := fmt.Sprintf("hmac id=\\"%s\\", algorithm=\\"hmac-sha1\\",
headers=\\"x-date\\", signature=\\"%s\\"",
        ApiAppKey, signature)

    // Construct the request
```

```
    headers := map[string]string{
        "Host":          Host,
        "Accept":        Accept,
        "Content-Type":  ContentType,
        "x-date":        xDate,
        "Authorization": sign,
    }

    // Send the request
    req, err := http.NewRequest(HTTPMethod, Url, strings.NewReader(bodyStr))
    if err != nil {
        log.Fatal(err)
    }

    for k, v := range headers {
        req.Header.Add(k, v)
    }

    res, err := http.DefaultClient.Do(req)
    if err != nil {
        log.Fatal(err)
    }
    defer res.Body.Close()

    resBody, _ := ioutil.ReadAll(res.Body)

    fmt.Println(string(resBody))
}
```

Sample code according to the hmac_sm3 algorithm

For more information, see Sample Project.

```
package main

import (
    "crypto/md5"
    "encoding/base64"
    "encoding/hex"
    "fmt"
    "io/ioutil"
    "log"
    "net/http"
    "net/url"
    "sort"
    "strings"
```

```
    "time"
    "unsafe"
)

/*
#cgo CFLAGS: -I./
#cgo LDFLAGS: -L./ -lTencentSM
#include "sm.h" // Quotation mark is used because it is not a standard "C"
header file
*/
import "C"


func mySM3_HMAC(data []byte, dataLen int, key []byte, keyLen int, mac []byte,
macLen int) int {
    if data == nil || key == nil || mac == nil || len(mac) != macLen {
        panic("invalid parameter")
    }
    return int(C.SM3_HMAC((*C.uchar)(unsafe.Pointer(&data[0])), (C.size_t)
(dataLen),
        (*C.uchar)(unsafe.Pointer(&key[0])), (C.size_t)(keyLen), (*C.uchar)
(unsafe.Pointer(&mac[0]))))
}

func main() {
    // Application's `ApiAppKey`
    const ApiAppKey = "Your ApiAppKey"
    // Application's `ApiAppSecret`
    const ApiAppSecret = "Your ApiAppSecret"

    const Url = "http://service-xxx-xxx.gz.apigw.tencentcs.com/a?c=1&b=3"

    const GmtFormat = "Mon, 02 Jan 2006 15:04:05 GMT"
    const HTTPMethod = "GET"
    const Accept = "application/json"
    const ContentType = "application/json"

    // Parse `Host` and `Path` according to `Url`
    u, err := url.Parse(Url)
    if err != nil {
        log.Fatal(err)
    }
    Host:= u.Hostname()
    Path:= u.Path
    Query := u.RawQuery

    // Environment information is not included in the signature path
```

```
    if strings.HasPrefix(Path, "/release") {
        Path = strings.TrimPrefix(Path, "/release")
    }else if strings.HasPrefix(Path, "/test") {
        Path = strings.TrimPrefix(Path, "/test")
    }else if strings.HasPrefix(Path, "/prepub") {
        Path = strings.TrimPrefix(Path, "/prepub")
    }

    if Path == "" {
        Path = "/"
    }

    // Splice query parameters. The query parameters need to be sorted in
lexicographical order.
    if len(Query) > 0 {
        args, _ := url.ParseQuery(Query)

        var keys []string
        for k := range args {
            keys = append(keys, k)
        }
        sort.Strings(keys)

        sortQuery := ""
        for _, k := range keys {
            if args[k][0] != "" {
                sortQuery = sortQuery + "&" + k + "=" + args[k][0]
            } else {
                sortQuery = sortQuery + "&" + k
            }
        }
        sortQuery = strings.TrimPrefix(sortQuery, "&")

        Path = Path + "?" + sortQuery
    }

    // Get the current UTC time
    xDate := time.Now().UTC().Format(GmtFormat)
    ContentMD5 := ""
    bodyStr := `{"arg1":"a","arg2":"b"}`
    if HTTPMethod == "POST" {
        h := md5.New()
        h.Write([]byte(bodyStr))
        md5Str := hex.EncodeToString(h.Sum(nil))
        ContentMD5 = base64.StdEncoding.EncodeToString([]byte(md5Str))
    }
```

```
    // Construct the signature
    signingStr := fmt.Sprintf("x-date: %s\\n%s\\n%s\\n%s\\n%s\\n%s", xDate,
HTTPMethod, Accept, ContentType,
        ContentMD5, Path)

    data := []byte(signingStr)
    key := []byte(ApiAppSecret)
    var out [32]byte

    mySM3_HMAC(data[:], len(data), key[:], len(key), out[:], len(out))
    signature := base64.StdEncoding.EncodeToString(out[:])

    sign := fmt.Sprintf("hmac id=\\"%s\\", algorithm=\\"hmac-sm3\\",
headers=\\"x-date\\", signature=\\"%s\\"",
        ApiAppKey, signature)

    // Construct the request
    headers := map[string]string{
        "Host":          Host,
        "Accept":        Accept,
        "Content-Type":  ContentType,
        "x-date":        xDate,
        "Authorization": sign,
    }

    // Send the request
    req, err := http.NewRequest(HTTPMethod, Url, strings.NewReader(bodyStr))
    if err != nil {
        log.Fatal(err)
    }

    for k, v := range headers {
        req.Header.Add(k, v)
    }

    res, err := http.DefaultClient.Do(req)
    if err != nil {
        log.Fatal(err)
    }
    defer res.Body.Close()

    resBody, _ := ioutil.ReadAll(res.Body)

    fmt.Println(string(resBody))
}
```

Java

Last updated：2023-12-22 10:06:26

Overview

This document describes how to manage access to your APIs through application authentication in Java.

Directions

1. In the API Gateway console, create an API and select the authentication type as **App authentication**. To learn more about the authentication types, find the documentation for different types of backends in Creating API - Overview.
2. Release the service where the API resides to an environment. See Service Release and Deactivation.
3. Create an application on the Application page in the console.
4. Select the created application in the application list, click **Bind API**, select the service and API, and click **Submit** to bind the application to the API.
5. Generate signing information in Java by referring to the Sample Code.

Environmental Dependencies

API Gateway provides sample codes with the request body in JSON format and form-data format. Select as needed. Application authentication in the Java demo needs to introduce the following external dependency:

```
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.5.13</version>
</dependency>
<dependency>
    <groupId>commons-codec</groupId>
    <artifactId>commons-codec</artifactId>
    <version>1.11</version>
</dependency>
```

Notes

For more information on operations such as application lifecycle management, authorizing an app to access the API, and binding an app to an API, see Application Management.
For the application signature generation process, see Application Authentication.

Sample Code

Sample code with the request body in JSON format

```
import org.apache.commons.codec.digest.DigestUtils;
import org.apache.http.HttpEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

import javax.crypto.Mac;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.*;

public class AppAuthJavaDemo {
    private static final String MAC_NAME = "HmacSHA1";
    private static final String ENCODING = "UTF-8";
    private static final String HTTP_METHOD_GET = "GET";
    private static final String HTTP_METHOD_POST = "POST";

    private static String getGMTTime(){
        Calendar cd = Calendar.getInstance();
        SimpleDateFormat sdf = new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss
'GMT'", Locale.US);
        sdf.setTimeZone(TimeZone.getTimeZone("GMT"));
        String GMTTime = sdf.format(cd.getTime());
        return GMTTime;
    }

    private static String sortQueryParams(String queryParam){
        // Parameters should be in alphabetical order
        if (queryParam == null || queryParam == ""){
            return "";
        }
```

```
        String[] queryParams = queryParam.split("&");
        Map<String, String> queryPairs = new TreeMap<>();
        for(String query: queryParams){
            String[] kv = query.split("=");
            queryPairs.put(kv[0], kv[1]);
        }

        StringBuilder sortedParamsBuilder = new StringBuilder();
        Iterator iter = queryPairs.entrySet().iterator();
        while(iter.hasNext()){
            Map.Entry entry = (Map.Entry) iter.next();
            sortedParamsBuilder.append(entry.getKey());
            sortedParamsBuilder.append("=");
            sortedParamsBuilder.append(entry.getValue());
            sortedParamsBuilder.append("&");
        }
        String sortedParams = sortedParamsBuilder.toString();
        sortedParams = sortedParams.substring(0, sortedParams.length() - 1);

        return sortedParams;
    }

    private static byte[] HmacSHA1Encrypt(String encryptText, String
encryptKey) throws Exception {
        byte[] data = encryptKey.getBytes(ENCODING);
        SecretKey secretKey = new SecretKeySpec(data, MAC_NAME);
        Mac mac = Mac.getInstance(MAC_NAME);
        mac.init(secretKey);

        byte[] text = encryptText.getBytes(ENCODING);
        return mac.doFinal(text);
    }

    private static String base64Encode(byte[] key) {
        final Base64.Encoder encoder = Base64.getEncoder();
        return encoder.encodeToString(key);
    }

    private static String getMD5(String str) {
        String md5Hex = DigestUtils.md5Hex(str);
        return md5Hex;
    }

    public static void main(String[] args) throws Exception {
        String environment = "";
```

```
        String url = "http://service-xxxxxxx-
xxxxxxxxxx.cq.apigw.tencentcs.com/appParam?name=clare&password=333";
        String host = "service-xxxxxxx-xxxxxxxxxx.cq.apigw.tencentcs.com";
        String apiAppKey = "APIDoMSRiefxxxxxxxxxxxxGz6AEEaFB";
        String apiAppSecret = "I0IDUmr6xxxxxxxxxxxxxx3C5GUsN2Rjvp";
        String httpMethod = "POST";
        String acceptHeader = "application/json";

        String contentType = "application/json";
        String reqBody = "
{\\"current\\":1,\\"size\\":10,\\"businessType\\":\\"4\\"}";
        String contentMD5 = base64Encode(getMD5(reqBody).getBytes());

        if (httpMethod.toUpperCase() == HTTP_METHOD_GET) {
            reqBody = "";
        }
        // ContentType and contentMd5 should be empty if request body is not
present
        if (reqBody.length() == 0) {
            contentType = "";
            contentMD5 = "";
            reqBody = "";
        }

        // Parse URL and assemble string to sign
        URL parsedUrl = new URL(url);
        String pathAndParams = parsedUrl.getPath();
        if (environment != ""){
            pathAndParams =
pathAndParams.substring(pathAndParams.indexOf(environment) +
environment.length());
        }

        if (parsedUrl.getQuery() != null) {
            pathAndParams = pathAndParams + "?" +
sortQueryParams(parsedUrl.getQuery());
        }

        System.out.println("pathAndParams:"+ pathAndParams);

        String xDate = getGMTTime();
        String stringToSign = String.format("x-date:
%s\\n%s\\n%s\\n%s\\n%s\\n%s", xDate, httpMethod, acceptHeader, contentType,
contentMD5, pathAndParams);
        // Encode string with HMAC and base64
        byte[] hmacStr = HmacSHA1Encrypt(stringToSign, apiAppSecret);
        String signature = base64Encode(hmacStr);
```

```
        String authHeader = String.format("hmac id=\\"%s\\", algorithm=\\"hmac-
sha1\\", headers=\\"x-date\\", signature=\\"%s\\"", apiAppKey, signature);

        System.out.println(stringToSign);
        // Generate request
        CloseableHttpClient httpClient = HttpClients.createDefault();
        CloseableHttpResponse response = null;
        // Send request
        if (httpMethod.toUpperCase() == HTTP_METHOD_GET){
            HttpGet httpGet = new HttpGet(url);
            httpGet.setHeader("Accept", acceptHeader);
            httpGet.setHeader("Host", host);
            httpGet.setHeader("x-date", xDate);
            httpGet.setHeader("Authorization", authHeader);
            response = httpClient.execute(httpGet);
        }

        if (httpMethod.toUpperCase() == HTTP_METHOD_POST) {
            HttpPost httpPost = new HttpPost(url);
            httpPost.setHeader("Accept", acceptHeader);
            httpPost.setHeader("Host", host);
            httpPost.setHeader("x-date", xDate);
            httpPost.setHeader("Content-Type", contentType);
            httpPost.setHeader("Content-MD5", contentMD5);
            httpPost.setHeader("Authorization", authHeader);
            StringEntity stringEntity = new StringEntity(reqBody, ENCODING);
            httpPost.setEntity(stringEntity);
            response = httpClient.execute(httpPost);
        }

        // Receive response
        HttpEntity responseEntity = response.getEntity();
        if (responseEntity != null) {
            System.out.println("Response status code: " +
response.getStatusLine());
            System.out.println("Response body: " +
EntityUtils.toString(responseEntity));
        }
    }
}
```

Sample code with the request body in form-data format

```
import org.apache.commons.codec.digest.DigestUtils;
import org.apache.http.HttpEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
```

```
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

import javax.crypto.Mac;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.*;

public class AppAuthJavaFormDemo {
    private static final String MAC_NAME = "HmacSHA1";
    private static final String ENCODING = "UTF-8";
    private static final String HTTP_METHOD_GET = "GET";
    private static final String HTTP_METHOD_POST = "POST";

    private static String getGMTTime(){
        Calendar cd = Calendar.getInstance();
        SimpleDateFormat sdf = new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss
'GMT'", Locale.US);
        sdf.setTimeZone(TimeZone.getTimeZone("GMT"));
        String GMTTime = sdf.format(cd.getTime());
        return GMTTime;
    }

    private static String sortQueryParams(String queryParam){
        // Parameters should be in alphabetical order
        if (queryParam == null || queryParam == ""){
            return "";
        }

        String[] queryParams = queryParam.split("&");
        Map<String, String> queryPairs = new TreeMap<>();
        for(String query: queryParams){
            String[] kv = query.split("=");
            queryPairs.put(kv[0], kv[1]);
        }

        StringBuilder sortedParamsBuilder = new StringBuilder();
        Iterator iter = queryPairs.entrySet().iterator();
        while(iter.hasNext()){
            Map.Entry entry = (Map.Entry) iter.next();
            sortedParamsBuilder.append(entry.getKey());
```

```
            sortedParamsBuilder.append("=");
            sortedParamsBuilder.append(entry.getValue());
            sortedParamsBuilder.append("&");
        }
        String sortedParams = sortedParamsBuilder.toString();
        sortedParams = sortedParams.substring(0, sortedParams.length() - 1);

        return sortedParams;
    }

    private static byte[] HmacSHA1Encrypt(String encryptText, String
encryptKey) throws Exception {
        byte[] data = encryptKey.getBytes(ENCODING);
        SecretKey secretKey = new SecretKeySpec(data, MAC_NAME);
        Mac mac = Mac.getInstance(MAC_NAME);
        mac.init(secretKey);

        byte[] text = encryptText.getBytes(ENCODING);
        return mac.doFinal(text);
    }

    private static String base64Encode(byte[] key) {
        final Base64.Encoder encoder = Base64.getEncoder();
        return encoder.encodeToString(key);
    }


    public static void main(String[] args) throws Exception {
        String environment = "";
        String url = "http://service-xxxxxxx-
xxxxxxxxxx.cq.apigw.tencentcs.com/appParam?name=clare&password=333";
        String host = "service-xxxxxxx-xxxxxxxxxx.cq.apigw.tencentcs.com";
        String apiAppKey = "APIDoMSRiefxxxxxxxxxxxxGz6AEEaFB";
        String apiAppSecret = "I0IDUmr6xxxxxxxxxxxxxx3C5GUsN2Rjvp";
        String httpMethod = "POST";
        String acceptHeader = "application/json";

        String reqBody = "";
        String contentType = "application/x-www-form-urlencoded";
        String contentMD5 = "";

        if (httpMethod.toUpperCase() == HTTP_METHOD_POST) {
            // Parse form data and assemble request body
            Map<String, String> reqBodyMap = new TreeMap<>();
            reqBodyMap.put("type", "fruit");
            reqBodyMap.put("fruitname", "apple");
```

```
            StringBuffer reqBodyBuffer = new StringBuffer();
            for (Map.Entry<String, String> e : reqBodyMap.entrySet()) {
                reqBodyBuffer.append(e.getKey());
                reqBodyBuffer.append("=");
                reqBodyBuffer.append(e.getValue());
                reqBodyBuffer.append("&");
            }
            reqBody = reqBodyBuffer.toString();
            reqBody = reqBody.substring(0, reqBody.length() - 1);
        }

        // ContentType should be empty if request body is not present
        if (reqBody.length() == 0) {
            contentType = "";
            reqBody = "";
        }

        // Parse URL and assemble string to sign
        URL parsedUrl = new URL(url);
        String pathAndParams = parsedUrl.getPath();
        if (environment != ""){
            pathAndParams =
pathAndParams.substring(pathAndParams.indexOf(environment) +
environment.length());
        }

        String queryParams = "";
        if (parsedUrl.getQuery() != null) {
            queryParams = parsedUrl.getQuery();
        }
        if (reqBody != "" && reqBody.length() > 0){
            if(queryParams.length() > 0){
                queryParams = queryParams + "&" + reqBody;
            } else {
                queryParams = reqBody;
            }
        }
        if (queryParams != ""){
            pathAndParams = pathAndParams + "?" + sortQueryParams(queryParams);
        }

        String xDate = getGMTTime();
        String stringToSign = String.format("x-date:
%s\\n%s\\n%s\\n%s\\n%s\\n%s", xDate, httpMethod, acceptHeader, contentType,
contentMD5, pathAndParams);
        System.out.println("stringToSign:" + stringToSign);
```

```
        // Encode string with HMAC and base64
        byte[] hmacStr = HmacSHA1Encrypt(stringToSign, apiAppSecret);
        String signature = base64Encode(hmacStr);
        String authHeader = String.format("hmac id=\\"%s\\", algorithm=\\"hmac-
sha1\\", headers=\\"x-date\\", signature=\\"%s\\"", apiAppKey, signature);

        // Generate request
        CloseableHttpClient httpClient = HttpClients.createDefault();
        CloseableHttpResponse response = null;

        // Send request
        if (httpMethod.toUpperCase() == HTTP_METHOD_GET) {
            HttpGet httpGet = new HttpGet(url);
            httpGet.setHeader("Accept", acceptHeader);
            httpGet.setHeader("Host", host);
            httpGet.setHeader("x-date", xDate);
            httpGet.setHeader("Authorization", authHeader);
            response = httpClient.execute(httpGet);
        }

        if (httpMethod.toUpperCase() == HTTP_METHOD_POST) {
            HttpPost httpPost = new HttpPost(url);
            httpPost.setHeader("Accept", acceptHeader);
            httpPost.setHeader("Host", host);
            httpPost.setHeader("x-date", xDate);
            httpPost.setHeader("Content-Type", contentType);
            httpPost.setHeader("Content-MD5", contentMD5);
            httpPost.setHeader("Authorization", authHeader);
            StringEntity stringEntity = new StringEntity(reqBody, ENCODING);
            httpPost.setEntity(stringEntity);
            response = httpClient.execute(httpPost);
        }

        // Receive response
        HttpEntity responseEntity = response.getEntity();

        if (responseEntity != null) {
            System.out.println("Response status code: " +
response.getStatusLine());
            System.out.println("Response body: " +
EntityUtils.toString(responseEntity));
        }
    }
}
```

Signature Tool

Last updated：2023-12-22 10:06:39

Overview

The API Gateway signature tool is a web tool provided by Tencent Cloud API Gateway to generate request signatures for key pair authentication APIs. You can set parameters on the signature tool page in the API Gateway console to generate a request signature.

Prerequisites

Postman has been downloaded and installed (download address).

Directions

Step 1. Enter the basic information for a signature

1. Log in to the API Gateway console.
2. In the left sidebar, click **Tool** > **Signature Tool** to go to the signature tool page.
3. In the **Basic info** module, click **Obtain**. The system will automatically fill in the start time and end time of the signature validity period.
4. Enter any value as the signature watermark value.

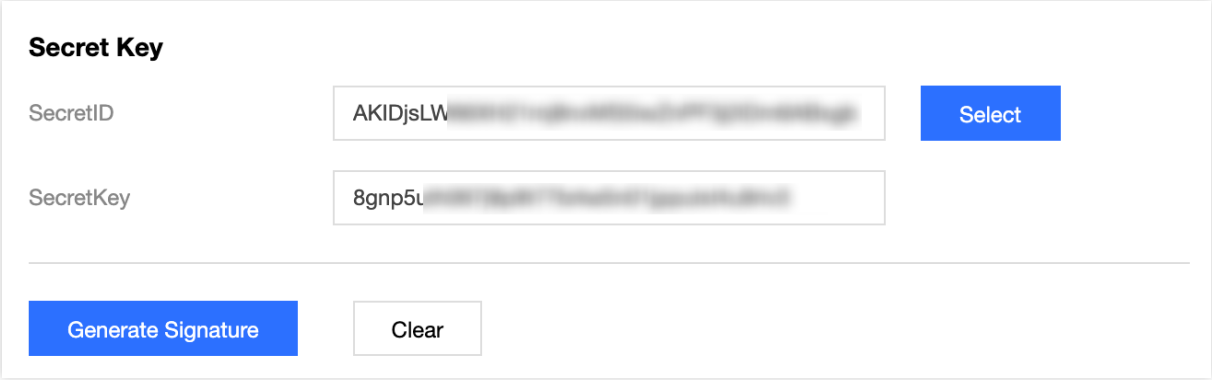
Step 2. Enter an API key pair

In the **Secret key** module, set the API key pair information using either of the following methods:

Method 1: click **Select** to select a key pair that has been created under your account.

Method 2: enter a key pair.

Make sure you enter the correct information. If the information is incorrect, your signature will be invalid.



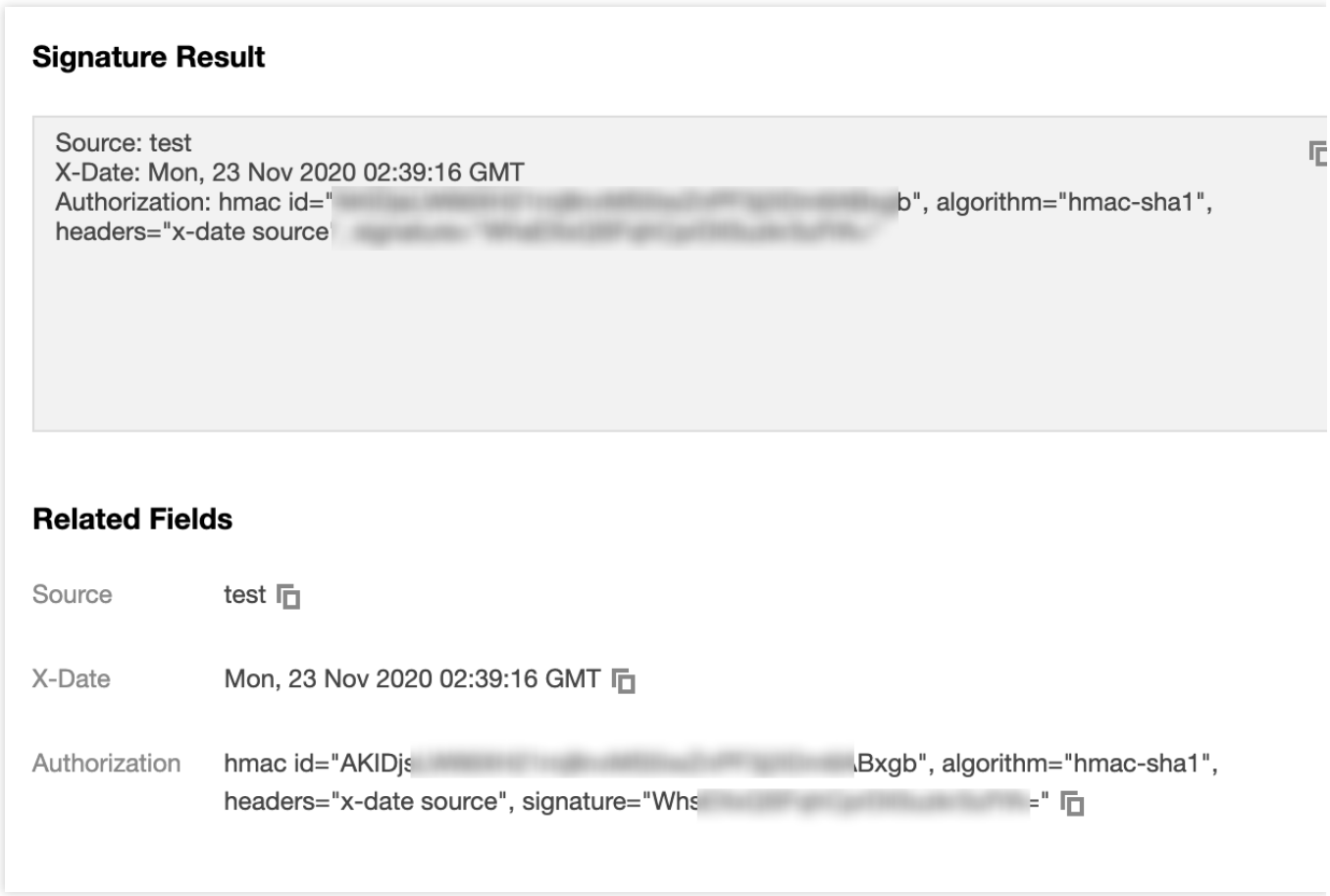
Step 3. Generate a signature

Click **Generate signature**. The request signature result will be displayed in the module on the right. The main parameters are as follows:

Source: signature watermark value.

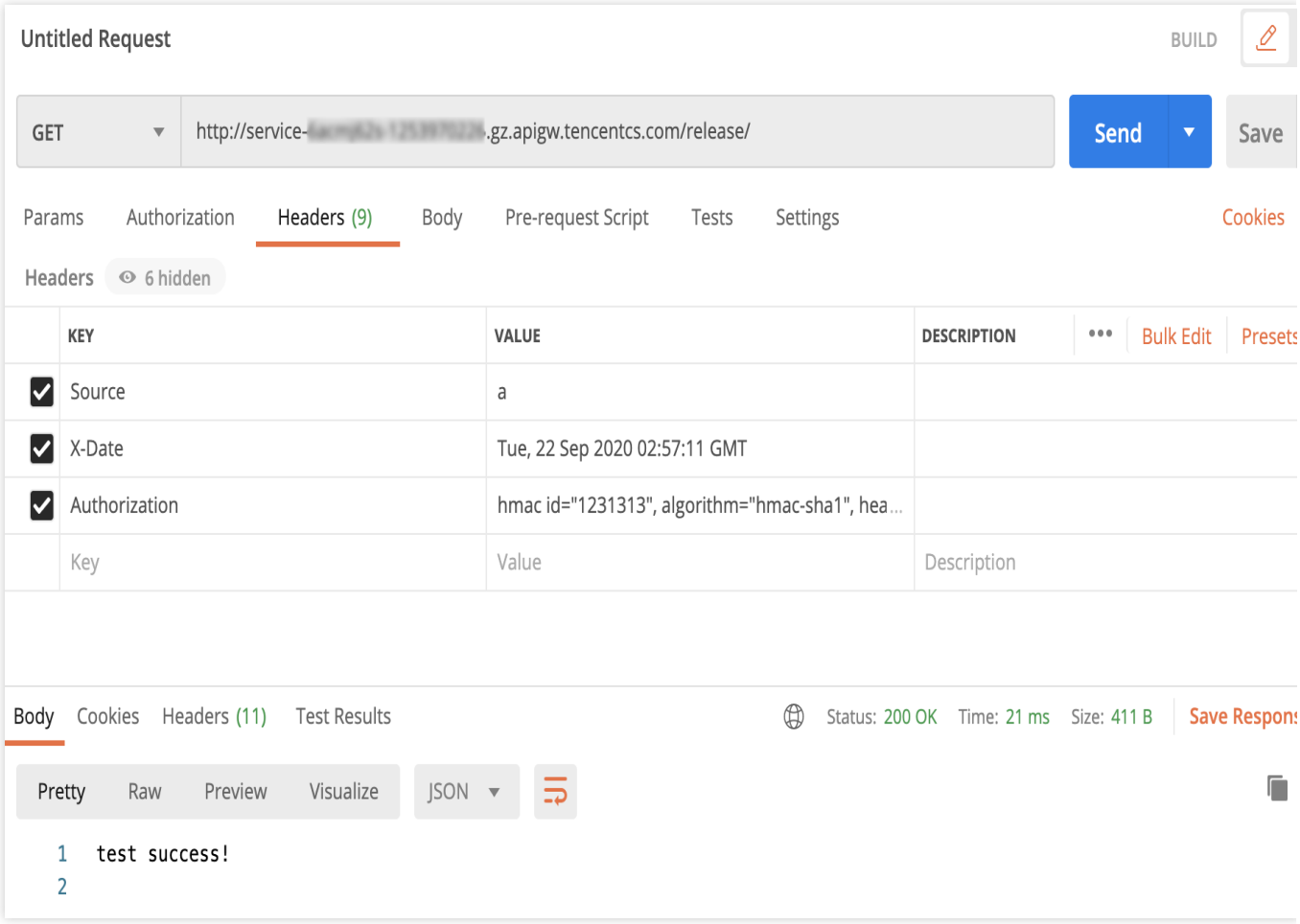
X-Date: HTTP request construction time in GMT format. The difference between **X-Date** and the current time cannot exceed 15 minutes.

Authorization: signing information.



Step 4. Initiate a call in Postman

Open Postman, enter the `Source`, `X-Date`, and `Authorization` parameters in `Headers`, enter the API request address, API request parameters, and other information, and click **Send** to call the key pair authentication API.



Note

Because the server will verify the time, you must initiate a call within the validity period of the signature. Otherwise, the call will fail.

The signature tool only verifies the validity of the parameters, but does not identify or inform you of the specific incorrect parameters.

For more information about key pair authentication, see Key Pair Authentication.

Quick Service Deletion Tool

Last updated：2023-12-22 10:06:49

Operation Scenarios

To avoid business losses caused by accidental service deletion, you cannot directly delete a service if it contains APIs or activated services. You have to deactivate the service first, query its API list, batch delete the service APIs, and then delete the service, which is a complex process.
To cope with this problem, Tencent Cloud API Gateway team and Tencent Cloud Serverless Framework team have launched the Serverless Cleaner to quickly delete the API gateway service.

Directions

1. Click here to download Serverless Cleaner.
2. Use CLI to go to the directory where Serverless Cleaner is located.
3. On the CLI, run the following commands to install dependencies.

```
$ npm install
```

4. After dependency installation, go to the directory where Serverless Cleaner is located, copy `config.example.js`, and rename it as `config.js`.
5. Modify `config.js` based on the information of the services to delete.

```
module.exports = {
 tencent: {
     // Resource region. You can enter the following resource region if the region
     region: 'ap-guangzhou',
     // Identity verification
     credentials: {
         SecretId: '',
         SecretKey: '',
     },
     apigwOptions: {
         // IDs of the services that you do not want to delete. Before you use Serv
         exclude: [],
         // IDs of the services that you want to delete. include has a higher prior
         include: [],
     },
     scfOptions: {
         // Specify this option when you delete SCF resources. When deleting only t
```

```
            exclude: [],
            include: [],
        },
    },
};
```

6. On the CLI, run the following commands to delete the API Gateway service.

```
$ npm run clean
```

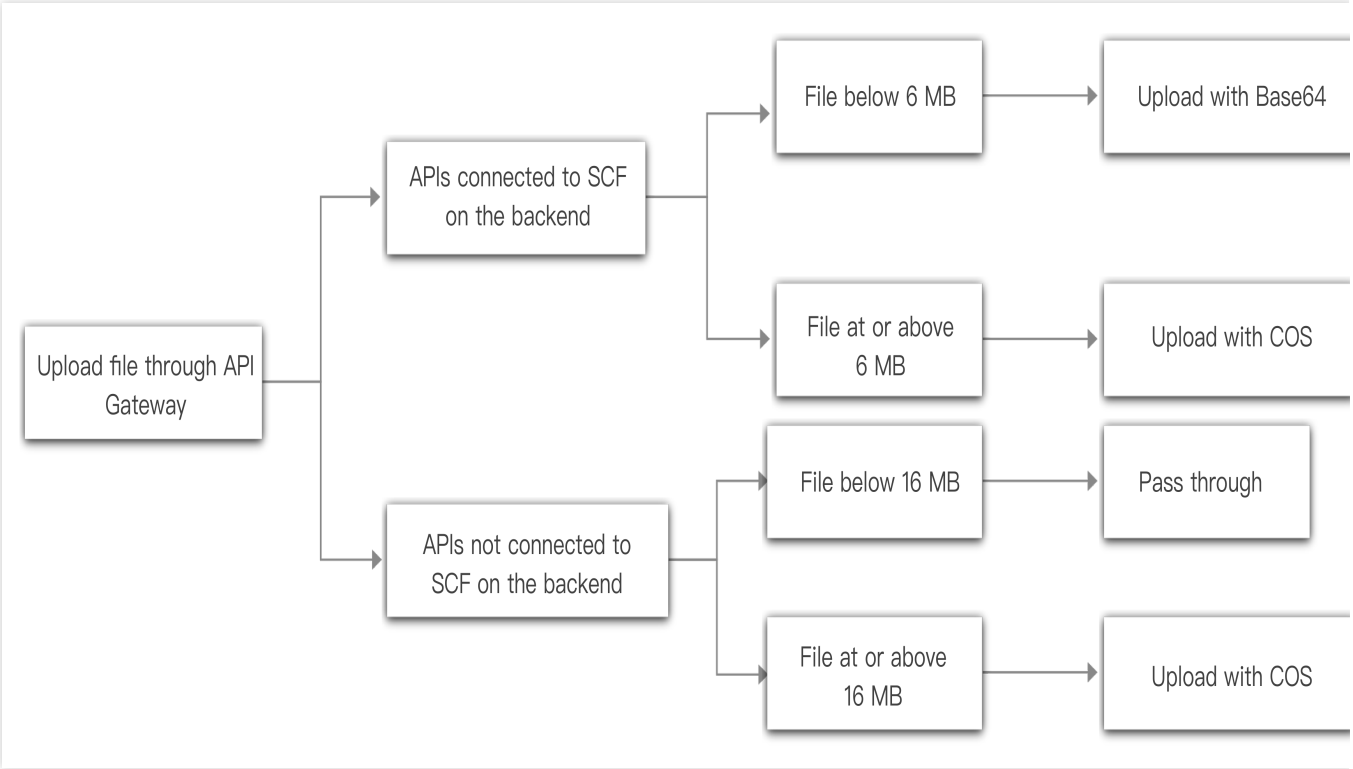
Uploading Files

Last updated：2023-12-22 10:06:59

Overview

This document describes how to upload files to the backend service through API Gateway. Based on different backend types and file sizes, there are the following four options, and you should choose the most appropriate one according to your actual business needs:

Upload Process



If the backend is connected to an SCF API to upload a Base64-encoded file below 6 MB in size, we recommend you Base64-encode the file through the client and pass the encoded content to SCF first. Then, SCF will Base64-decode the content to complete the upload.
If the backend is connected to an SCF API to upload a Base64-encoded file at or above 6 MB in size, we recommend you upload the file to COS through the client and pass the object address to SCF first. Then, SCF will pull the file from COS to complete the upload.

If the backend is connected to an API of another service to upload a file below 16 MB in size, you only need to leave the parameters empty and keep everything pass-through.
If the backend is connected to an API of another service to upload a file at or above 16 MB in size, we recommend you upload the file to COS through the client and pass the object address to the backend service first. Then, the backend service will pull the file from COS to complete the upload.

Serverless Multi-File Upload Processing

Last updated：2023-12-22 10:07:09

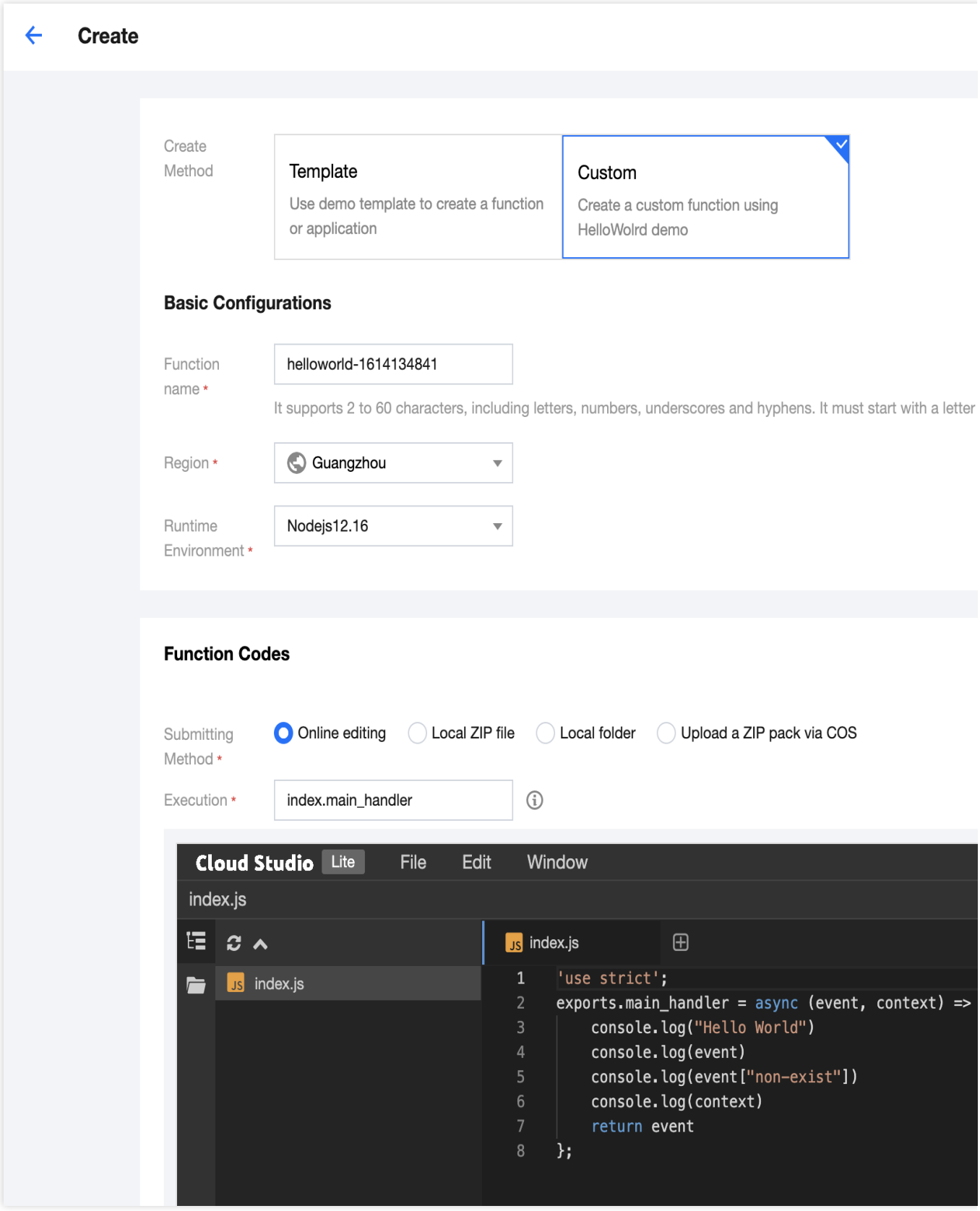
Overview

To process an HTTP request for `multipart/form-data` multi-file upload through Tencent Cloud Serverless, the Base64 encoding capability of API Gateway is needed to encode the `multipart` byte stream in the original HTTP request into a string, so that the HTTP event can be serialized and passed to SCF for processing.
After SCF gets and Base64-decodes the `body` in the `event` passed in by API Gateway, the generated byte stream is the same as that in a regular HTTP request and can be processed normally. In Node.js, it can be processed with libraries such as `busboy` .

Directions

Step 1. Create a function

1. Log in to the SCF console.
2. Click **Function Service** in the left sidebar. On the page that appears, click **Create** to create a `Node.js` function.
The specific parameters for creation are as follows:



3. Click **Complete**.

Step 2. Write and deploy code

1. After creating the function, you can refer to the following sample code to write the specific logic for processing `multipart/form-data`.

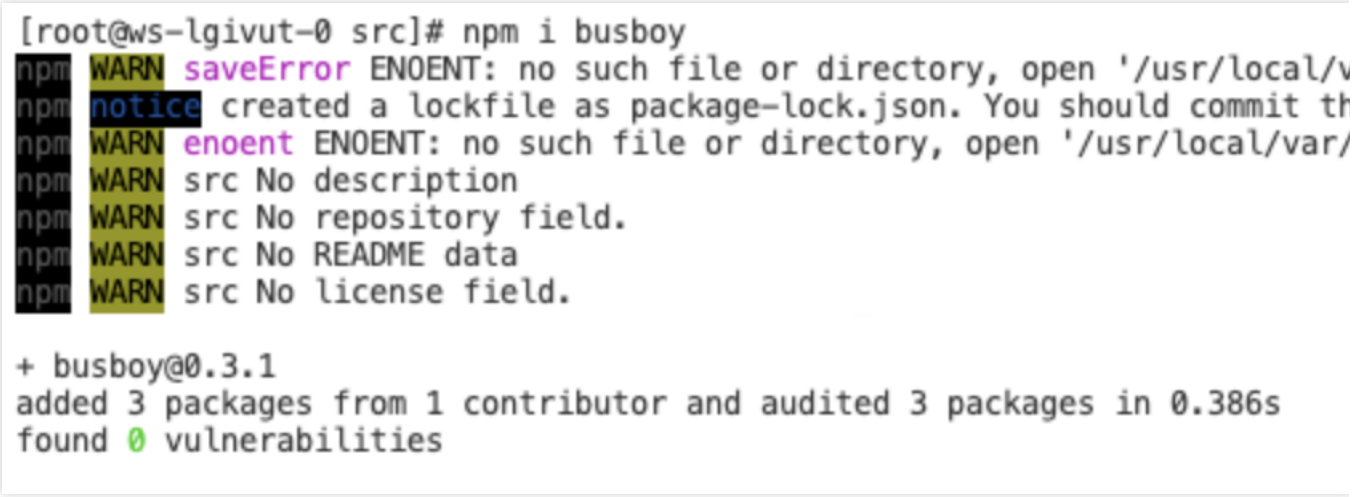
```
// handler.js
"use strict";
const stream = require("stream");
const Busboy = require("busboy");

/** User upload (POST) is processed */
const handlePost = (event) => {
  return new Promise((resolve, reject) => {
    const busboy = new Busboy({ headers: event.headers });
    let html = "";
    /** The file is received */
    busboy.on("file", (fieldname, file, filename, encoding, mimetype) => {
      let buf = Buffer.alloc(0);
      console.log({ fieldname });
      /** File data blocks are received and spliced into a complete buffer */
      file.on("data", function (data) {
        buf = Buffer.concat([buf, data]);
      });
      /** File data block receipt is completed, and the DOM string is generated
*/
      file.on("end", function () {
        const imgBase64 = buf.toString("base64");
        html += `![](data:${mimetype};base64, ${imgBase64})`;
      });
    });
    /** multipart/form-data receipt is completed, and the generated HTML is
constructed and returned */
    busboy.on("finish", function () {
      console.log({ msg: "Parse form complete!", html });
      resolve({
        statusCode: 200,
        headers: {
          "content-type": "text/html",
        },
        body: html,
      });
    });

    /**
     * busboy needs to process the data in the form of stream pipe.
     * After the body is decoded to the buffer,
     * it is converted into a stream and finally piped to busboy
     */
    const bodyBuf = Buffer.from(event.body, "base64");
```

```
    var bufferStream = new stream.PassThrough();
    bufferStream.end(bodyBuf);
    bufferStream.pipe(busboy);
  });
};

/** The static file is returned */
const handleGet = (event) => {
  const html = `<html><head></head><body>
    <form method="POST" enctype="multipart/form-data">
    <input type="file" name="image-1" accept="image/*"><br />
     <input type="file" name="image-2" accept="image/*"><br />
     <input type="submit">
    </form>
    </body></html>`;
  console.log({ msg: "Get form complete!", html });
  return {
    statusCode: 200,
    headers: {
      "content-type": "text/html",
    },
    body: html,
  };
};

/** Function entry */
exports.main_handler = async (event, context) => {
  const method = event.httpMethod;
  /** If the request is a POST request, the user's multipart/form-data is
processed and a page displaying the upload result is generated */
  if (method === "POST") {
    return handlePost(event);
  }
  /** If the request is a GET request, the page where the file is uploaded is
returned */
  if (method === "GET") {
    return handleGet(event);
  }
};
```

2. After writing the code, you can also install the runtime dependencies for the function. For example, you can use `busboy` to decode the `multipart/form-data` data.

Note:

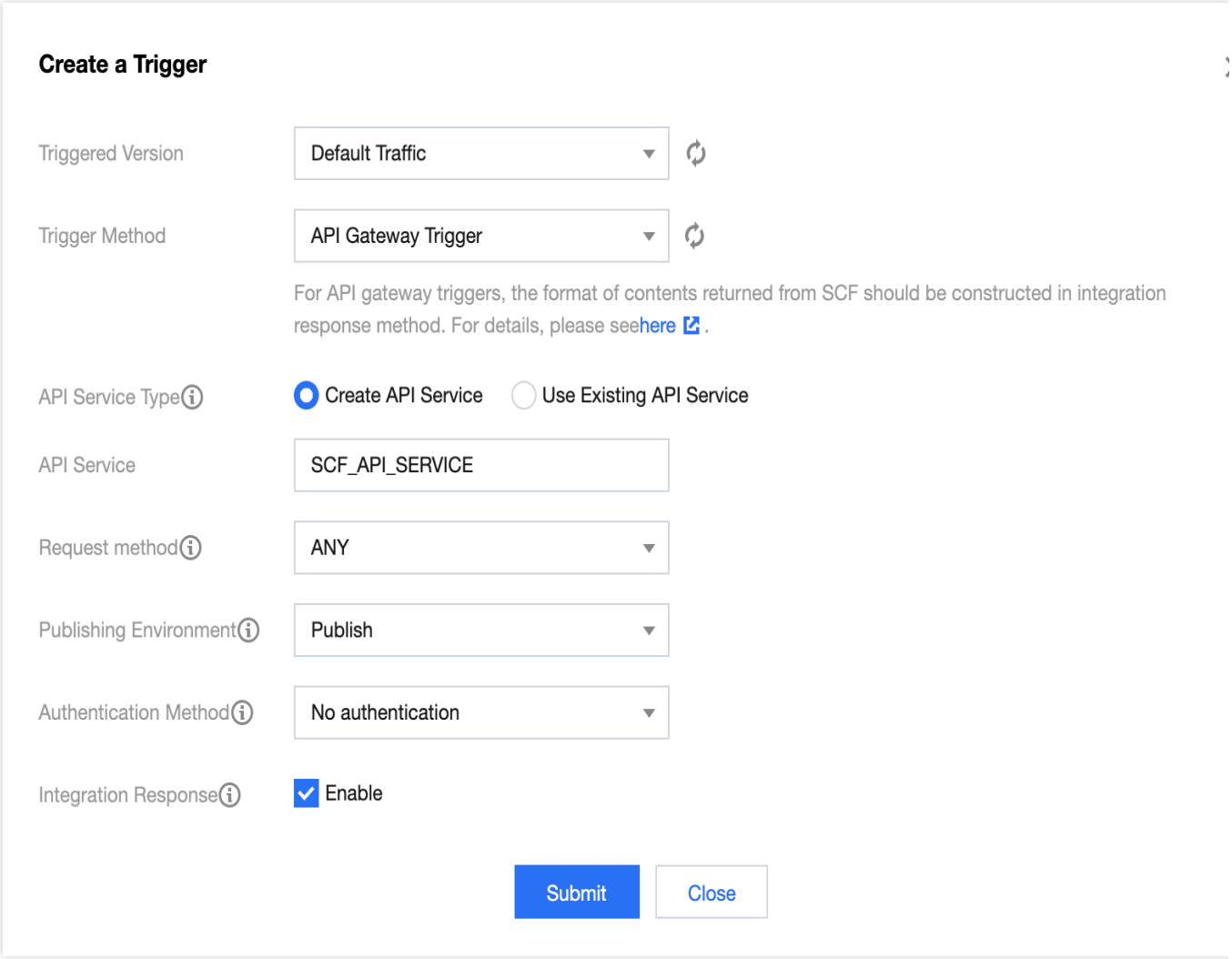
Dependencies must be installed in the `src` folder.



3. Click **Deploy** to complete the function deployment.

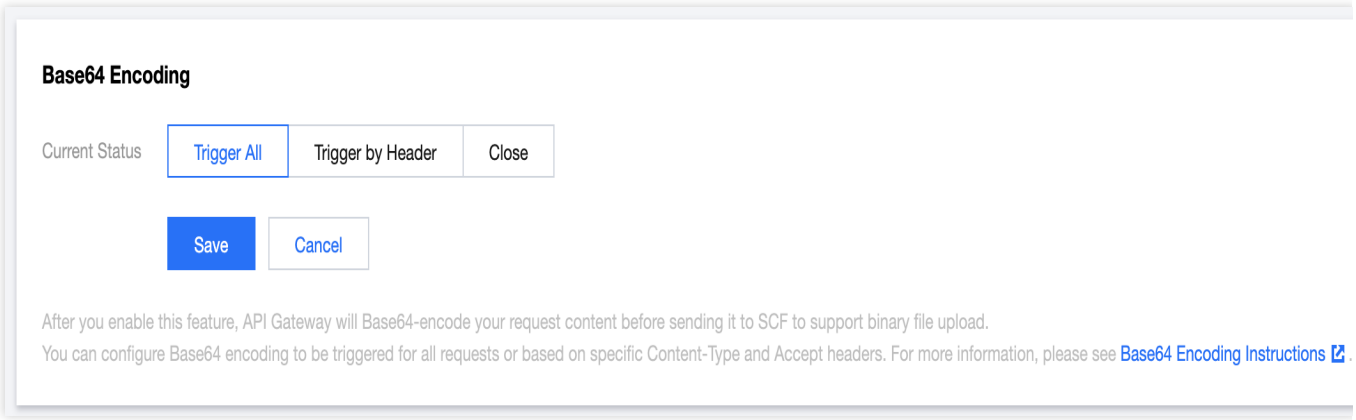
Step 3. Bind an API Gateway trigger

In the trigger management of the function, you need to bind an API Gateway trigger to the function before it can process users' specific HTTP requests. The specific binding method and configuration are as follows:



At this time, if you access the link bound by API Gateway, you will find that although the static page can work, the page does not display the correct result after an image is uploaded. This is because the Base64 encoding feature is disabled in API Gateway by default. As a result, the `multipart/form-data` data is incorrectly encoded as a string and passed to the handler function, and busboy cannot decode it.

Therefore, you need to enter API Gateway, find the bound API service, and enable Base64 encoding in the basic configuration.



After the service is opened and released, it can work normally.

Demo

You can access the official demo built by Tencent Cloud Serverless to view the effect of file upload.

Structures Passed by API Gateway to Backend

Last updated：2023-12-22 10:07:20

Overview

When a client sends requests to API Gateway, API Gateway will process the requested content and then pass it to the backend. In this case, the structures that API Gateway passes to the backend are generally fixed.
This document describes the structures passed by API Gateway to different types of backends to facilitate development and troubleshooting.

Structure for Public URLs/IP and VPC Resource Backends

```
GET / HTTP/1.1
Host: 10.0.0.0
Connection: keep-alive
X-Client-Proto: http
X-Client-Proto-Ver: HTTP/1.1
X-Forwarded-For: 100.100.10.1
X-Real-IP: 100.100.10.1
User-Agent: curl/7.29.0
Accept: /
x-b3-traceid: 12345678906f******7cd8db0fe843dc
X-Api-RequestId: 12345678906f******7cd8db0fe843dc
X-Api-Scheme: http
```

The data structure is described as follows:

Element	Description
Host	Specifies the virtual host.
Connection	Determines whether to keep the network open when the current request is completed.
X-Client-Proto	Records client request protocol.
X-Client-Proto-Ver	Records client request protocol version.
X-Forwarded-	Records every IP address accessed after the request is initiated by the client.

For	
X-Real-IP	Records the real IP address of client request.
User-Agent	Records client request agent.
Accept	Represents the type of data the client wants to accept.
x-b3-traceid	Records the `RequestId` of this request. It is equivalent to `X-Api-RequestId` and is used to adapt to the internal module of API Gateway.
X-Api-RequestId	Records the `RequestId` of this request.
X-Api-Scheme	Records the client request protocol. It is equivalent to `X-Client-Proto` and is used to adapt to the internal module of API Gateway.

Note:

As the request body structure varies depending on `Content-Type` and the client request may have custom headers, this document lists only the fixed request headers passed by API Gateway.

Structure for SCF Backends

```
{
    "body": "{key:value}",
    "requestContext": {
        "httpMethod": "ANY",
        "serviceId": "service-dlhbuxqh",
        "path": "/scf/{pathParam}",
        "sourceIp": "14.17.22.36",
        "identity": {},
        "stage": "release"
    },
    "queryStringParameters": {
        "queryParam": ""
    },
    "headers": {
        "content-length": "11",
        "x-b3-traceid": "12345678906f******7cd8db0fe843dc",
        "x-qualifier": "$DEFAULT",
        "accept": "/",
        "user-agent": "curl/7.69.1",
        "host": "service-abcdefgh-1234567890.gz.apigw.tencentcs.com",
        "requestsource": "APIGW",
        "x-api-scheme": "http",
```

```
        "x-api-requestid": " 12345678906f******7cd8db0fe843dc",
        "content-type": "application/x-www-form-urlencoded"
    },
    "pathParameters": {
        "pathParam": "mypath"
    },
    "queryString": {
        "a": "1",
        "b": "2"
    },
    "httpMethod": "POST",
    "headerParameters": {
        "headerparam": ""
    },
    "path": "/scf/mypath",
    "isBase64Encoded": "true",
}
```

The data structure is described as follows:

Element	Description
requestContext	Configuration information, request ID, authentication information, and source information about API Gateway where the request comes from. `serviceId` , `path` , and `httpMethod` are the service ID, API path, and method of API Gateway, respectively. `stage` indicates the environment of the request source API. `requestId` indicates the unique ID of the current request. `identity` indicates the user's authentication method and information. `sourceIp` identifies the request source IP.
path	Records the complete path of the actual request.
httpMethod	Records the HTTP method of the actual request.
queryString	Records the complete query content of the actual request.
body	Records the content of the actual request after being converted into a string.
headers	Records the complete headers of the actual request.
pathParameters	Records the path parameters configured in API Gateway and their actual values.
queryStringParameters	Records the query parameters configured in API Gateway and their actual values.
headerParameters	Records the header parameters configured in API Gateway and their actual values.
isBase64Encoded	Records whether the request body is Base64-encoded. Valid values: `true` ,

	`false`