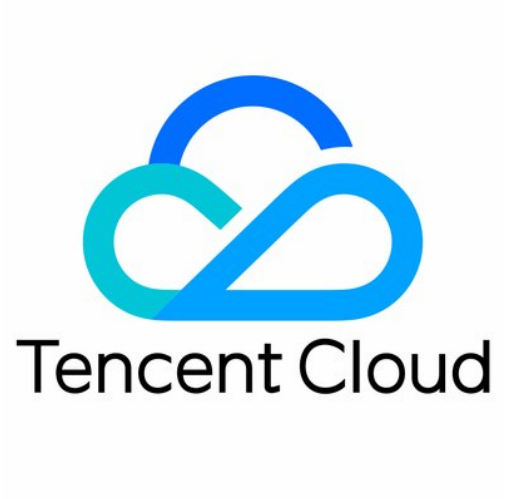
API Gateway

Legacy Features

Product Documentation



Contents

Legacy Features
Key Pair Authentication
Key Management

Last updated：2023-12-22 10:07:58

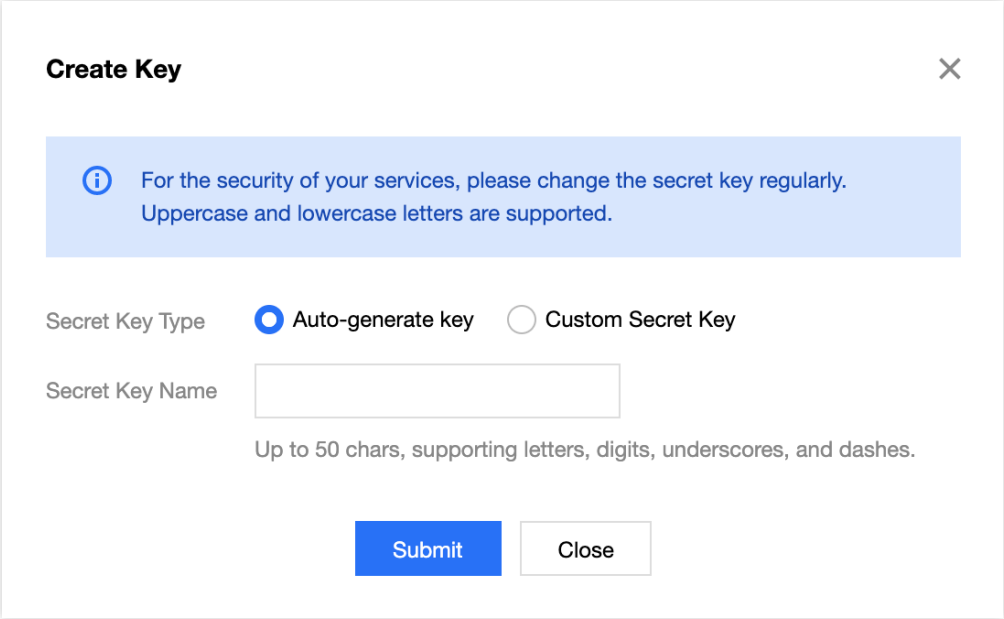
Overview

This document describes how to manage keys in the API Gateway console.
Creating a key: you can create a key using the **auto-generate key** or the **custom secret key** method. A successfully created key must be bound to an API using a usage plan so that the key can be used as a credential for accessing the API.
Disabling/enabling a key: after a key is disabled, the API Gateway rejects all requests with signatures generated by using the key. After the key is enabled again, the API Gateway can properly process requests with signatures generated by using the key.
Changing a key: you can reset the SecretKey of a key while the secret key name and SecretId remain unchanged.
Deleting a key: you can delete a key in the **Disabled** state. A key in the **In Use** state cannot be deleted.

Directions

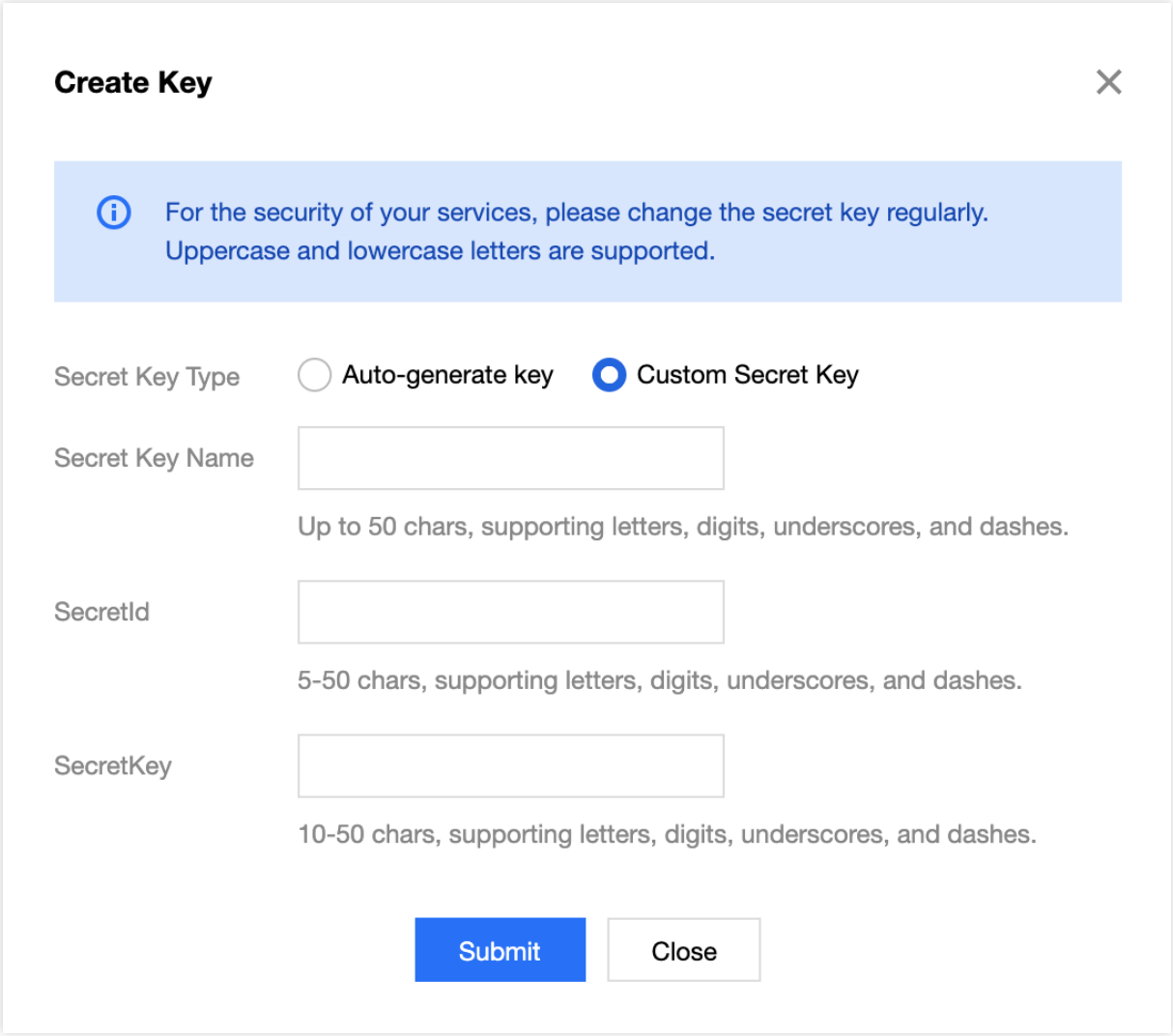
Auto-generating a key

1. Log in to the API Gateway console.
2. In the left sidebar, click **Key** to go to the key list page.
3. Click **Create** to open the **Create Key** dialog box.
4. Set **Secret Key Type** to **Auto-generate key** and enter a secret key name.
5. Click **Submit** to auto-generate a key.



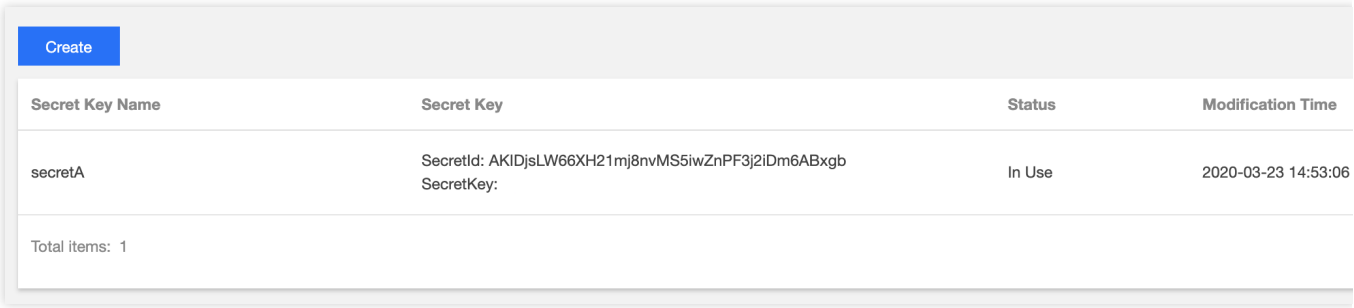
Creating a custom key

1. Log in to the API Gateway console.
2. In the left sidebar, click **Key** to go to the key list page.
3. Click **Create** to open the **Create Key** dialog box.
4. Set **Secret Key Type** to **Custom Secret Key** and enter a secret key name, SecretId, and SecretKey.
5. Click **Submit** to create a custom key.



Disabling a key

1. Log in to the API Gateway console.
2. In the left sidebar, click **Key** to go to the key list page.
3. Locate the target key in the **In Use** state and click **Disable**.
4. In the pop-up confirmation dialog box, click **Confirm** to disable the key. After that, the status of the key changes to **Disabled**.



Enabling a key

1. Log in to the API Gateway console.
2. In the left sidebar, click **Key** to go to the key list page.
3. Locate the target key in the **Disabled** state and click **Enable**.
4. In the pop-up confirmation dialog box, click **Confirm** to enable the key. After that, the status of the key will change to **In Use**.

Changing a key

1. Log in to the API Gateway console.
2. In the left sidebar, click **Key** to go to the key list page.
3. Locate the target key in the **In Use** state and click **Change**.
4. In the pop-up confirmation dialog box, click **Confirm** to change the key; that is, to reset the SecretKey of the key.

Deleting a key

1. Log in to the API Gateway console.
2. In the left sidebar, click **Key** to go to the key list page.
3. Locate the target key in the **Disabled** state and click **Delete**.
4. In the pop-up confirmation dialog box, click **Confirm** to delete the key.

Notes

A key in the **Disabled** state cannot be changed or bound to a usage plan. To perform these operations, enable the key first.

A key in the **In Use** state cannot be deleted. To delete the key, disable it first.

The SecretId of a custom key is unique across all regions. If you fail to create a custom key, change the SecretId and try again.

Key Pair Authentication

Last updated：2023-12-22 10:08:08

You can use `secret_id` and `secret_key` for authentication management of your APIs. As `secret_id` and `secret_key` appear in pairs, they are called `secret_id/secret_key` pair.
You need to create a secret_id/secret_key pair before the authentication.

Key Pair Authentication for API

When creating an API, you can choose the authentication type as "key pair authentication". Then, after the API is released, only access requests to it initiated with the correct key pair can pass the authentication by API Gateway, while requests without the key or correct key pair cannot.
After the API is created, you need to use the "usage plan" feature to bind the key pair to the API or the service where the API resides. For more information on the configuration, please see Usage Plan.

Key Content

Sample **secret_id**: `AKIDCg*****j548pN` . It is used to identify which key is used, participates in signature computation, and is reflected in the transfer process.
Sample **secret_key**: `ZxF2wh*****N2oPrC` . It is used for signature computation, but is not reflected in the transfer process.

Signature Computation

Content to be delivered at last

The HTTP request delivered at last contains at least two headers: Date/X-Date and Authorization. More headers such as source in a request are also workable.
The value of Date header is the time constructed by HTTP request in GMT, for example: Fri, 09 Oct 2015 00:00:00 GMT.
The value of X-Date header is the time constructed by HTTP request in GMT, for example: Mon, 19 Mar 2018 12:08:40 GMT. The value for timeout is 15 minutes.
Source header represents the signature watermark value. You can enter any value for it or leave it empty.
The `Authorization` header is in the format of `Authorization: hmac id="secret_id", algorithm="hmac-sha1", headers="date source", signature="Base64(HMAC-`

`SHA1(signing_str, secret_key))"` .

The various parts of Authorization are explained as follows:

hmac: a fixed part used to indicate the computation method.

ID: the value of secret_id in the key.

algorithm: the encryption algorithm. hmac-sha1 is supported now.

headers: refer to headers that involve in signature computation, sorting by the order of actual computation.

signature: the signature obtained after signature computation is completed, with `signing_str` as its content.

Signature computation method

A signature has 2 parts and is computed according to the specified encryption algorithm. Take hmac-sha1 algorithm as an example:

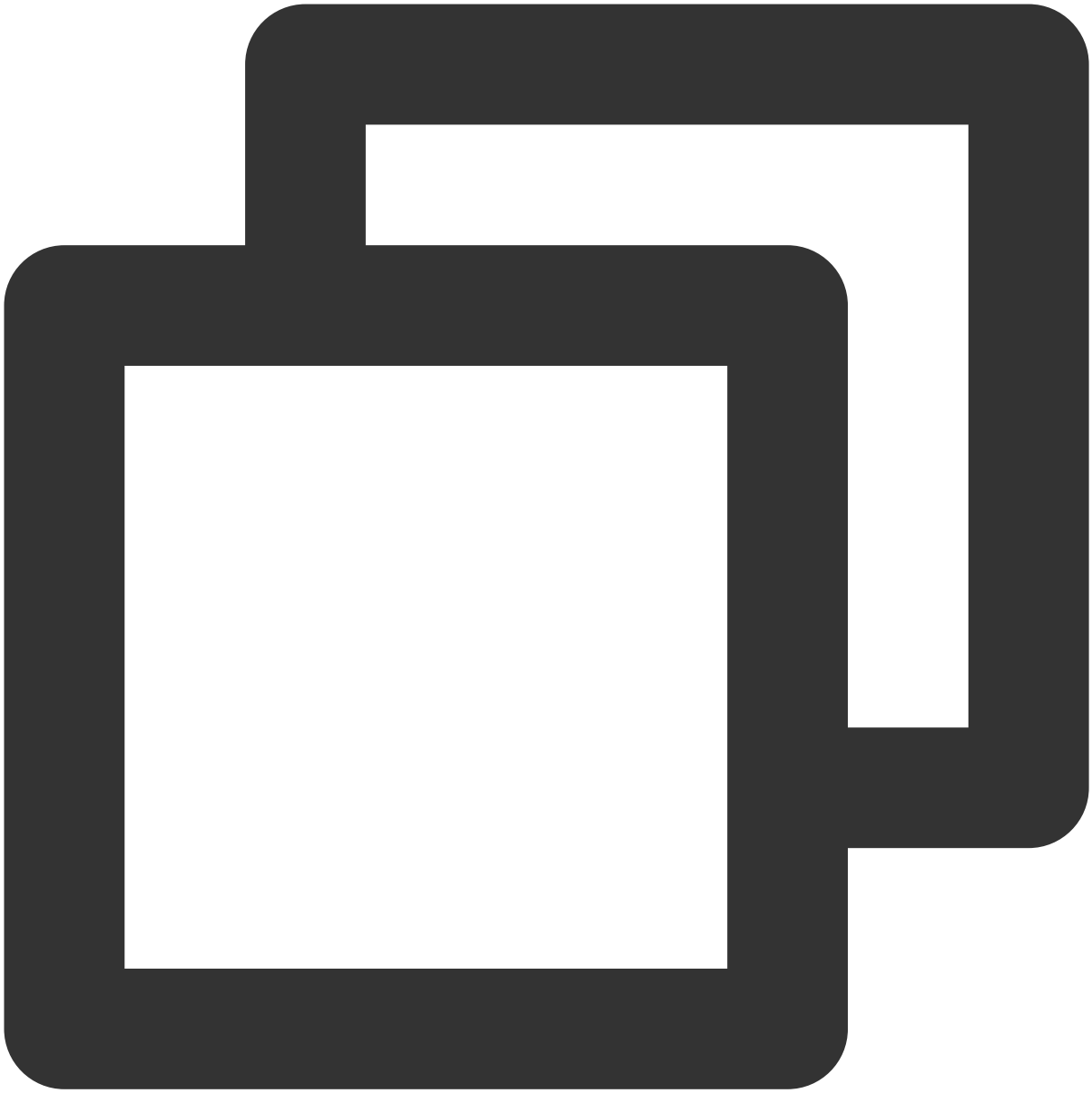
Signature content

First, generate the signature content, which consists of custom headers. It is recommended to include "date" at least in the header. But you can also include more headers.

Headers are converted according to the following requirements and then sorted in sequence:

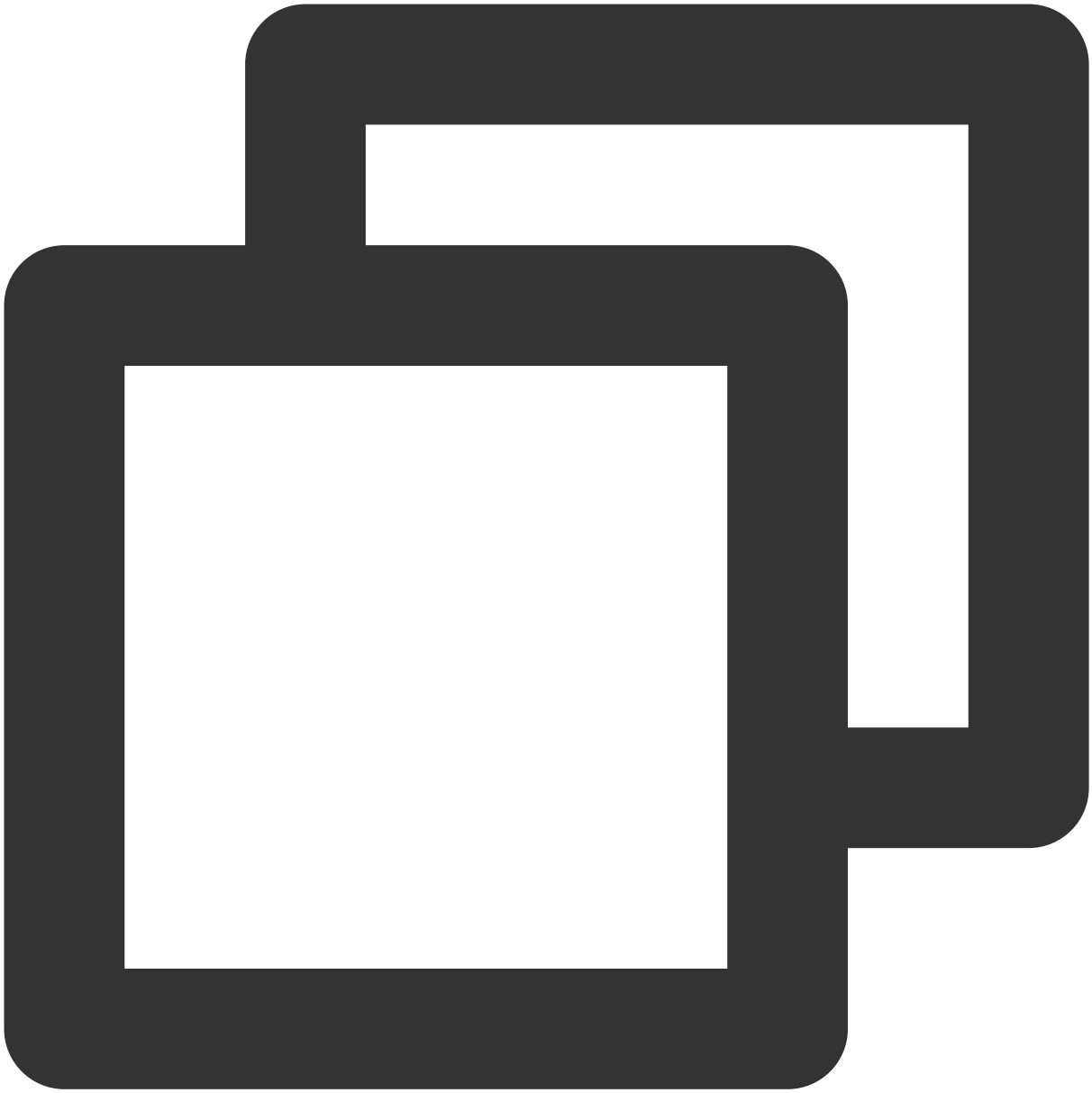
1. The header name is converted to lowercase and followed by the **ASCII character** `:` and then **an ASCII space character**.
2. Attach the value of the header.
3. If it is not the last header that needs to construct a signature, attach **ASCII new-line character** `\\n` .

For example, if there are two headers involved in constructing the signature content (this example is for your reference only. Please fill in the fields as needed):



```
Date:Fri, 09 Oct 2015 00:00:00 GMT
Source:AndriodApp
```

The generated signature content is as follows:



```
date: Fri, 09 Oct 2015 00:00:00 GMT
source: AndriodApp
```

Computing a signature

Base64(HMAC-SHA1(signing_str, secret_key)) algorithm is used to compute the signature content generated in the previous step to generate a signature, that is:
Using the signature content as input information, and secret_key content as the key, compute according to HMAC-SHA1 algorithm to get the encrypted signature content.

Convert the computed encrypted signature content to deliverable signature content using Base64.

Using a signature

As shown in **Content to be delivered at last**, at "signature" in the Authorization header, enter the signature computed in the last step.

Notes

header matching

The headers in `Authorization` are the ones involved in signature computation. We recommend you convert them to lowercase and separate them by ASCII spaces. For example, if the headers involved in the computation are `date` and `source` , the format should be `headers="date source"` ; if only the `x-date` header participates in the computation, the format should be `headers="x-date"` .

Signature content generation

When arranging the content, please pay extra attention to the colon and space after the header name. If they are missing, the verification may fail. `SecretId` , `SecretKey` , URL, and `Host` should be replaced with your real information.

Note:

For signature demos for common programming languages, please see Developer Guide - Generating Signatures in Multiple Programming Languages.

Java (Key Pair Authentication)

Last updated：2023-12-22 10:08:19

Operation Scenarios

This document describes how to authenticate and manage your APIs through key pair authentication in Java.

Directions

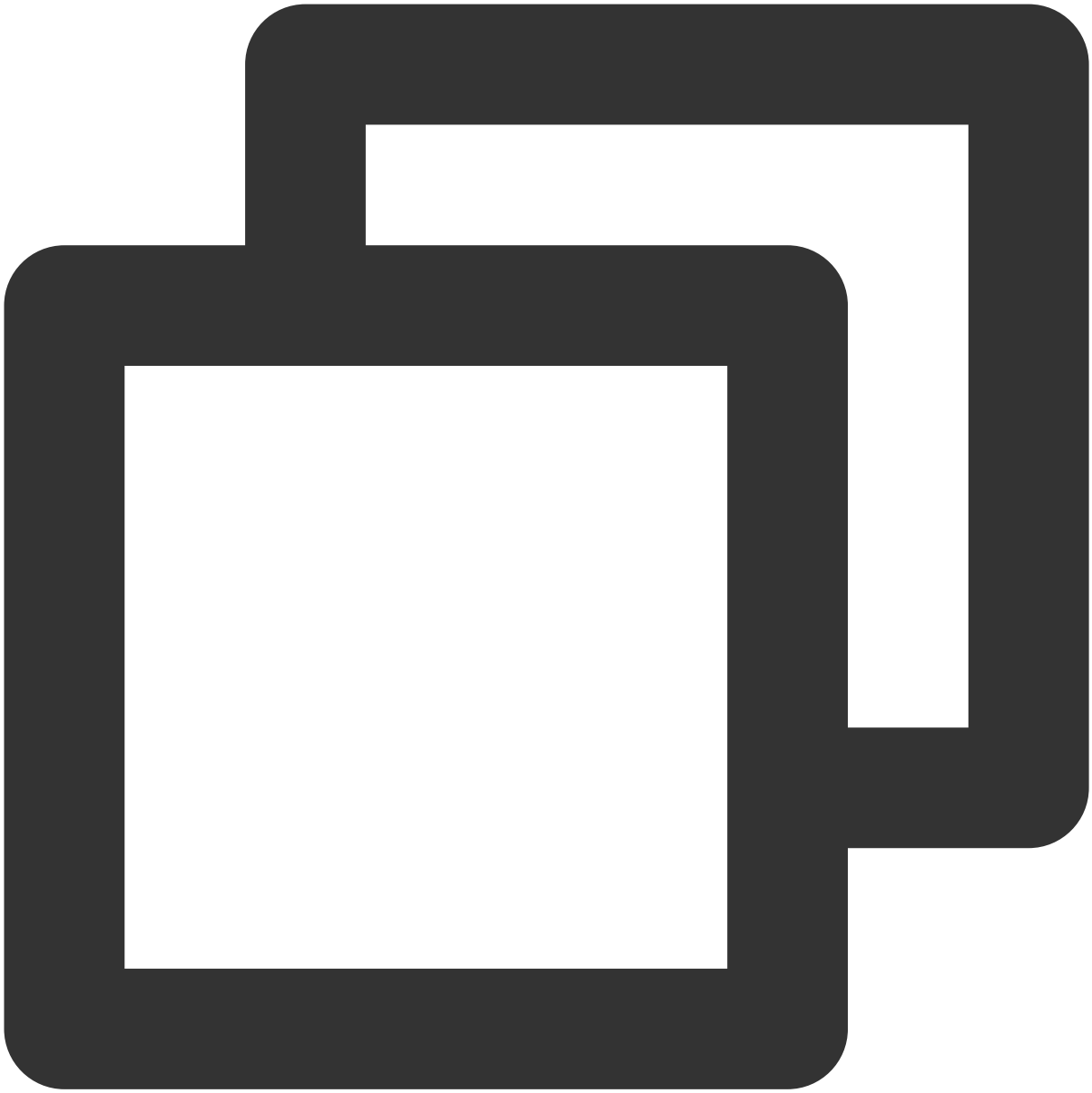
1. In the API Gateway Console, create an API and select the authentication type as "key pair authentication" (for more information, please see API Creation Overview).
2. Publish the service where the API resides to the release environment (for more information, please see Service Release and Deactivation).
3. Create a key pair on the key management page in the console.
4. Create a usage plan on the usage plan page in the console and bind it to the created key pair (for more information, please see Sample Usage Plan).
5. Bind the usage plan to the API or the service where the API resides.
6. Generate signing information in Java by referring to the Sample Code.

Notes

The eventually delivered HTTP request contains at least two headers: `Date` or `X-Date` and `Authorization`. More optional headers can be added in the request. If `Date` is used, the server will not check the time; if `X-Date` is used, the server will check the time.
The value of `Date` header is the construction time of the HTTP request in GMT format, such as Fri, 09 Oct 2015 00:00:00 GMT.
The value of `X-Date` header is the construction time of the HTTP request in GMT format, such as Mon, 19 Mar 2018 12:08:40 GMT. It cannot deviate from the current time for more than 15 minutes.
If it is a microservice API, you need to add two fields in the header: `X-NameSpace-Code` and `X-MicroService-Name`. They are not needed for general APIs and are included in the demo by default.
This Demo contains samples of the `GET` and `POST` methods for your choice.

Sample Code

Base64.java

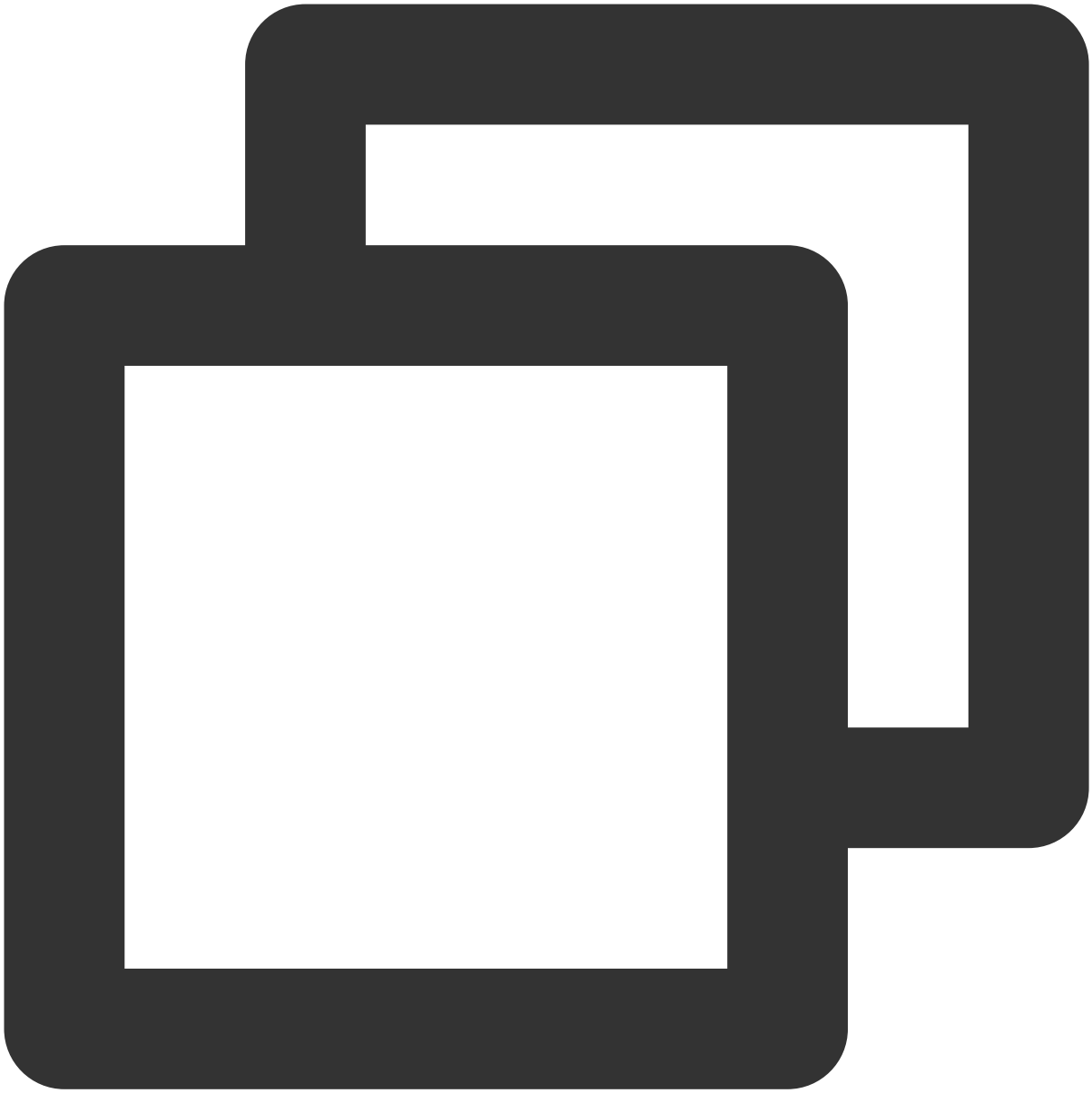


```
package apigatewayDemo;

import java.io.UnsupportedEncodingException;

public class Base64 {
    private static char[] base64EncodeChars = new char[] { 'A', 'B', 'C', 'D',
            'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q',
            'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd',
            'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q',
```

```
            'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '0', '1', '2', '3',
            '4', '5', '6', '7', '8', '9', '+', '/' };

    private static byte[] base64DecodeChars = new byte[] { -1, -1, -1, -1, -1,
            -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
            -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
            -1, -1, -1, -1, 62, -1, -1, -1, 63, 52, 53, 54, 55, 56, 57, 58, 59,
            60, 61, -1, -1, -1, -1, -1, -1, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
            10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, -1,
            -1, -1, -1, -1, -1, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37,
            38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, -1, -1, -1,
            -1, -1 };

    public static String encode(byte[] data) {
        StringBuffer sb = new StringBuffer();
        int len = data.length;
        int i = 0;
                                int b1, b2, b3;
        while (i < len) {
            b1 = data[i++] & 0xff;
            if (i == len) {
                sb.append(base64EncodeChars[b1 >>> 2]);
                sb.append(base64EncodeChars[(b1 & 0x3) << 4]);
                sb.append("==");
                break;
            }
            b2 = data[i++] & 0xff;
            if (i == len) {
                sb.append(base64EncodeChars[b1 >>> 2]);
                sb.append(base64EncodeChars[((b1 & 0x03) << 4)
                        | ((b2 & 0xf0) >>> 4)]);
                sb.append(base64EncodeChars[(b2 & 0x0f) << 2]);
                sb.append("=");
                break;
            }
            b3 = data[i++] & 0xff;
            sb.append(base64EncodeChars[b1 >>> 2]);
            sb.append(base64EncodeChars[((b1 & 0x03) << 4)
                    | ((b2 & 0xf0) >>> 4)]);
            sb.append(base64EncodeChars[((b2 & 0x0f) << 2)
                    | ((b3 & 0xc0) >>> 6)]);
                        sb.append(base64EncodeChars[b3 & 0x3f]);
        }
        return sb.toString();
    }

    public static byte[] decode(String str) throws UnsupportedEncodingException {
```

```
        StringBuffer sb = new StringBuffer();
        byte[] data = str.getBytes("US-ASCII");
        int len = data.length;
        int i = 0;
        int b1, b2, b3, b4;
        while (i < len) {
            /* b1 */
            do {
                b1 = base64DecodeChars[data[i++]];
            } while (i < len && b1 == -1);
            if (b1 == -1)
                break;
            /* b2 */
            do {
                b2 = base64DecodeChars[data[i++]];
            } while (i < len && b2 == -1);
            if (b2 == -1)
                break;
            sb.append((char) ((b1 << 2) | ((b2 & 0x30) >>> 4)));
            /* b3 */
            do {
                b3 = data[i++];
                if (b3 == 61)
                    return sb.toString().getBytes("ISO-8859-1");
                b3 = base64DecodeChars[b3];
            } while (i < len && b3 == -1);
            if (b3 == -1)
                break;
            sb.append((char) (((b2 & 0x0f) << 4) | ((b3 & 0x3c) >>> 2)));
            /* b4 */
            do {
                b4 = data[i++];
                if (b4 == 61)
                    return sb.toString().getBytes("ISO-8859-1");
                b4 = base64DecodeChars[b4];
            } while (i < len && b4 == -1);
            if (b4 == -1)
                break;
            sb.append((char) (((b3 & 0x03) << 6) | b4));
        }
        return sb.toString().getBytes("ISO-8859-1");
    }
}
```

SignAndSend.java



```
package apigatewayDemo;
import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.text.SimpleDateFormat;
import java.util.*;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
```

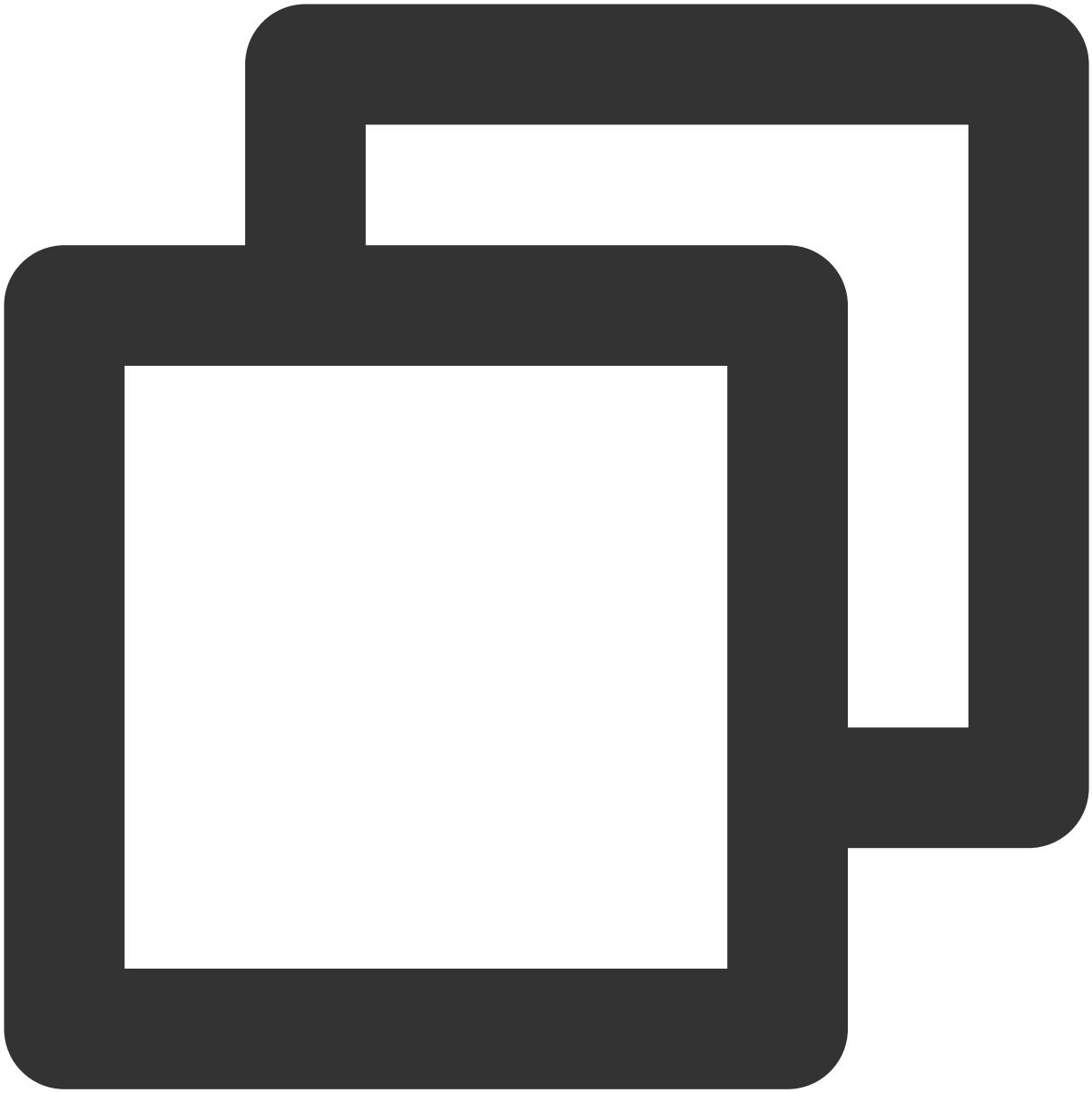
```
import java.io.PrintWriter;
import java.net.URL;
import java.net.URLConnection;
import java.net.HttpURLConnection;
import java.io.OutputStreamWriter;

public class SignAndSend {
    private static final String CONTENT_CHARSET = "UTF-8";
    private static final String HMAC_ALGORITHM = "HmacSHA1";
    public static String sign(String secret, String timeStr)
            throws NoSuchAlgorithmException, UnsupportedEncodingException, InvalidK
    {
        // Get the signature string
        String signStr = "date: "+timeStr+"\\n"+"source: "+"source";
        // Get the API signature
        String sig = null;
        Mac mac1 = Mac.getInstance(HMAC_ALGORITHM);
        byte[] hash;
        SecretKeySpec secretKey = new SecretKeySpec(secret.getBytes(CONTENT_CHARSET
        mac1.init(secretKey);
        hash = mac1.doFinal(signStr.getBytes(CONTENT_CHARSET));
        sig = new String(Base64.encode(hash));
        System.out.println("signValue--->" + sig);
        return sig;
    }

    public static HttpURLConnection NewHttpUrlCon(String url, String secretId, Stri
        // Get the current GMT time
        Calendar cd = Calendar.getInstance();
        SimpleDateFormat sdf = new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss 'GMT
        sdf.setTimeZone(TimeZone.getTimeZone("GMT"));
        String timeStr = sdf.format(cd.getTime());
        HttpURLConnection httpUrlCon = null;
        try {
            String urlNameString = url;
            URL realUrl = new URL(urlNameString);
            // Open the connection to the URL
            URLConnection connection = realUrl.openConnection();
            httpUrlCon = (HttpURLConnection)connection;
            // Set general request attributes
            httpUrlCon.setRequestProperty("Host", url);
            httpUrlCon.setRequestProperty("Accept", "text/html, */*; q=0.01");
            httpUrlCon.setRequestProperty("Source","source");
            httpUrlCon.setRequestProperty("Date",timeStr);
            String sig = sign(secretKey,timeStr);
            String authen = "hmac id=\\""+secretId+"\\", algorithm=\\"hmac-sha1\\",
            System.out.println("authen --->" + authen);
```

```
            httpUrlCon.setRequestProperty("Authorization",authen);
            httpUrlCon.setRequestProperty("X-Requested-With","XMLHttpRequest");
            httpUrlCon.setRequestProperty("Accept-Encoding","gzip, deflate, sdch");

            // If it is a microservice API, you need to add two fields in the heade
            httpUrlCon.setRequestProperty("X-NameSpace-Code","testmic");
            httpUrlCon.setRequestProperty("X-MicroService-Name","provider-demo");
        } catch (Exception e) {
            System.out.println("An exception occurred while creating the connection
            e.printStackTrace();
        }
        return httpUrlCon;
    }

    public static String sendGet(String url, String secretId, String secretKey) {
        String result = "";
        BufferedReader in = null;
        try {
            // Create a connection
            HttpURLConnection httpUrlCon = NewHttpUrlCon(url, secretId, secretKey);

            // Establish the actual connection
            httpUrlCon.connect();

            // Get all response header fields
            Map<String, List<String>> map = httpUrlCon.getHeaderFields();

            // Traverse all response header fields
            for (String key : map.keySet()) {
                System.out.println(key + "--->" + map.get(key));
            }

            // Define the `BufferedReader` input stream to read the URL response
            in = new BufferedReader(new InputStreamReader(
                    httpUrlCon.getInputStream()));
            String line;
            while ((line = in.readLine()) != null) {
                result += line;
            }
        } catch (Exception e) {
            System.out.println("An exception occurred while sending the GET request
            e.printStackTrace();
        }

        // Close the input stream by using the `finally` block
        finally {
            try {
```

```
                if (in != null) {
                    in.close();
                }
            } catch (Exception e2) {
                e2.printStackTrace();
            }
        }
        return result;
    }

    public static String sendPost(String url, String secretId, String secretKey) {
        String result = "";
        BufferedReader in = null;
        try {

            // Create a connection
            HttpURLConnection httpUrlCon = NewHttpUrlCon(url, secretId, secretKey);

            // Configure the `POST` request
            httpUrlCon.setRequestMethod("POST");
            httpUrlCon.setUseCaches(false);  // A `POST` cannot use the cache
            httpUrlCon.setDoOutput(true);
            httpUrlCon.setDoInput(true);

            // Put the request data in the body
            OutputStreamWriter out = new OutputStreamWriter(httpUrlCon.getOutputStr
            String jsonStr = "{\\"caller\\":\\"apigw\\", \\"data\\":\\"test post\\"
            out.write(jsonStr);
            out.flush();
            out.close();

            // Establish the actual connection
            httpUrlCon.connect();

            // Get all response header fields
            Map<String, List<String>> map = httpUrlCon.getHeaderFields();

            // Traverse all response header fields
            for (String key : map.keySet()) {
                System.out.println(key + "--->" + map.get(key));
            }

            // Define the `BufferedReader` input stream to read the URL response
            in = new BufferedReader(new InputStreamReader(
                    httpUrlCon.getInputStream()));
            String line;
            while ((line = in.readLine()) != null) {
```

```
                result += line;
            }
        } catch (Exception e) {
            System.out.println("An exception occurred while sending the POST reques
            e.printStackTrace();
        }

        // Close the input stream by using the `finally` block
        finally {
            try {
                if (in != null) {
                    in.close();
                }
            } catch (Exception e2) {
                e2.printStackTrace();
            }
        }
        return result;
    }
}
```

Demo.java



```
package apigatewayDemo;
public class Demo {
    public static void main(String[] args) {
        String secretId = "your secretId"; // `SecretId` in key pair
        String secretKey = "your secretKey"; // `SecretKey` in key pair
        SignAndSend signAndSendInstance = new SignAndSend();

        // `GET` request
        String getUrl = "http://service-xxxxxxxx-1234567890.gz.apigw.tencentcs.com:
        String getResult = SignAndSend.sendGet(getUrl, secretId, secretKey);
        System.out.println(getResult);
```

```
        // `POST` request
        String postUrl = "http://service-xxxxxxxx-1234567890.gz.apigw.tencentcs.com
        String postResult = SignAndSend.sendPost(postUrl, secretId, secretKey);
        System.out.println(postResult);
    }
}
```

Go (Key Pair Authentication)

Last updated：2023-12-22 10:08:28

Operation Scenarios

This document describes how to authenticate and manage your APIs through key pair authentication in Go.

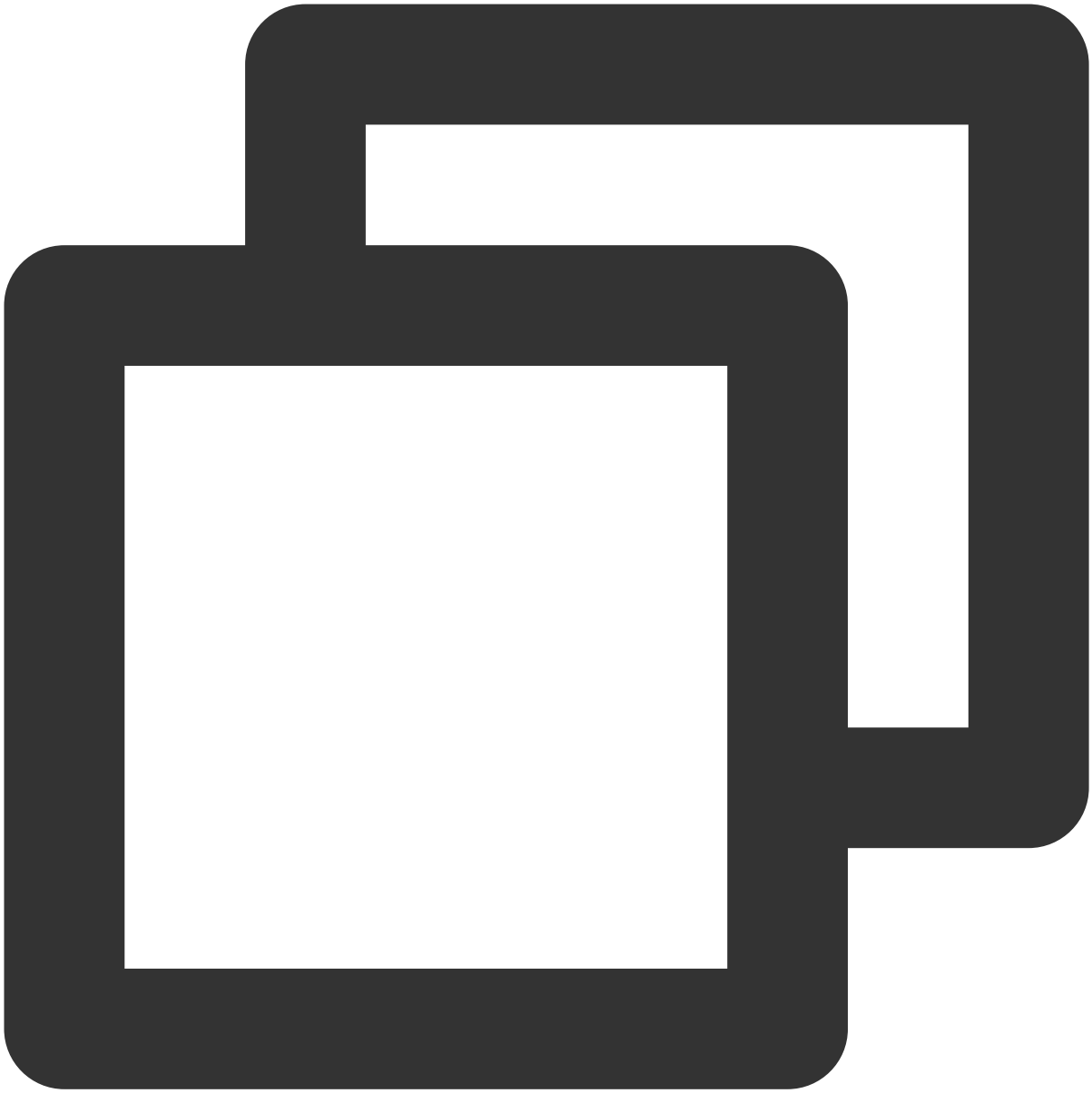
Directions

1. In the API Gateway Console, create an API and select the authentication type as "key pair authentication" (for more information, please see API Creation Overview).
2. Publish the service where the API resides to the release environment (for more information, please see Service Release and Deactivation).
3. Create a key pair on the key management page in the console.
4. Create a usage plan on the usage plan page in the console and bind it to the created key pair (for more information, please see Sample Usage Plan).
5. Bind the usage plan to the API or the service where the API resides.
6. Generate signing information in Go by referring to the Sample Code.

Notes

The eventually delivered HTTP request contains at least two headers: `Date` or `X-Date` and `Authorization`. More optional headers can be added in the request. If `Date` is used, the server will not check the time; if `X-Date` is used, the server will check the time.
The value of `Date` header is the construction time of the HTTP request in GMT format, such as Fri, 09 Oct 2015 00:00:00 GMT.
The value of `X-Date` header is the construction time of the HTTP request in GMT format, such as Mon, 19 Mar 2018 12:08:40 GMT. It cannot deviate from the current time for more than 15 minutes.
If it is a microservice API, you need to add two fields in the header: `X-NameSpace-Code` and `X-MicroService-Name`. They are not needed for general APIs and are included in the demo by default.

Sample Code



```
package main

import (
    "time"
    "fmt"
    "crypto/hmac"
    "crypto/sha1"
    "io"
    "io/ioutil"
    "encoding/base64"
    "net/http"
```

```
)


func calcAuthorization(source string, secretId string, secretKey string) (sign stri
    timeLocation, _ := time.LoadLocation("Etc/GMT")
    dateTime = time.Now().In(timeLocation).Format("Mon, 02 Jan 2006 15:04:05 GMT")
    sign = fmt.Sprintf("x-date: %s\\nsource: %s", dateTime, source)
    fmt.Println(sign)

    //hmac-sha1
    h := hmac.New(sha1.New, []byte(secretKey))
    io.WriteString(h, sign)
    sign = fmt.Sprintf("%x", h.Sum(nil))
    sign = string(h.Sum(nil))
    fmt.Println("sign:", fmt.Sprintf("%s", h.Sum(nil)))

    //base64
    sign = base64.StdEncoding.EncodeToString([]byte(sign))
    fmt.Println("sign:", sign)

    auth := fmt.Sprintf("hmac id=\\"%s\\", algorithm=\\"hmac-sha1\\", headers=\\"x-
        secretId, sign)
    fmt.Println("auth:", auth)

    return auth, dateTime, nil
}

func main () {
    SecretId := "your SecretId" // `SecretId` in key pair
    SecretKey := "your SecretKey" // `SecretKey` in key pair
    source := "xxxxxx" // Arbitrary signature watermark value

    sign, dateTime, err := calcAuthorization(source, SecretId, SecretKey)
    if err != nil {
        fmt.Println(err)
        return
    }

    defaultDomain := "service-xxxxxxxx-1234567890.ap-guangzhou.apigateway.myqcloud.

    reqUrl := "https://service-xxxxxxxx-1234567890.ap-guangzhou.apigateway.myqcloud
    client := &http.Client{
        Timeout: 7 * time.Second,//set timeout
    }

    req, err := http.NewRequest("GET", reqUrl, nil) //set body
    if err != nil {
```

```
        fmt.Println(err)
        return
    }

    req.Header.Set("Accept", "*/*")
    req.Header.Set("Accept-Charset", "utf-8;")
    req.Header.Set("Host", defaultDomain)
    req.Header.Set("Source", source)
    req.Header.Set("X-Date", dateTime)
    req.Header.Set("Authorization", sign)

    // If it is a microservice API, you need to add two fields in the header: 'X-Na
    req.Header.Set("x-NameSpace-Code", "testmic")
    req.Header.Set("x-MicroService-Name", "provider-demo")

    resp, err := client.Do(req)
    if err != nil {
        fmt.Println(err)
        return
    }
    defer resp.Body.Close()

    fmt.Println("status code:", resp.StatusCode)

    //get resp header
    var headerMsg string
    for key, _ := range resp.Header {
        headerMsg += fmt.Sprintf("\\n%s:%s", key, resp.Header.Get(key))
    }
    fmt.Println(headerMsg)

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        fmt.Println(err)
        return
    }

    fmt.Println(string(body))

}
```

Python (Key Pair Authentication)

Last updated：2023-12-22 10:08:38

Operation Scenarios

This document describes how to authenticate and manage your APIs through key pair authentication in Python.

Directions

1. In the API Gateway Console, create an API and select the authentication type as "key pair authentication" (for more information, please see API Creation Overview).
2. Publish the service where the API resides to the release environment (for more information, please see Service Release and Deactivation).
3. Create a key pair on the key management page in the console.
4. Create a usage plan on the usage plan page in the console and bind it to the created key pair (for more information, please see Sample Usage Plan).
5. Bind the usage plan to the API or the service where the API resides.
6. Generate signing information in Python by referring to the Sample Code.

Environment Dependencies

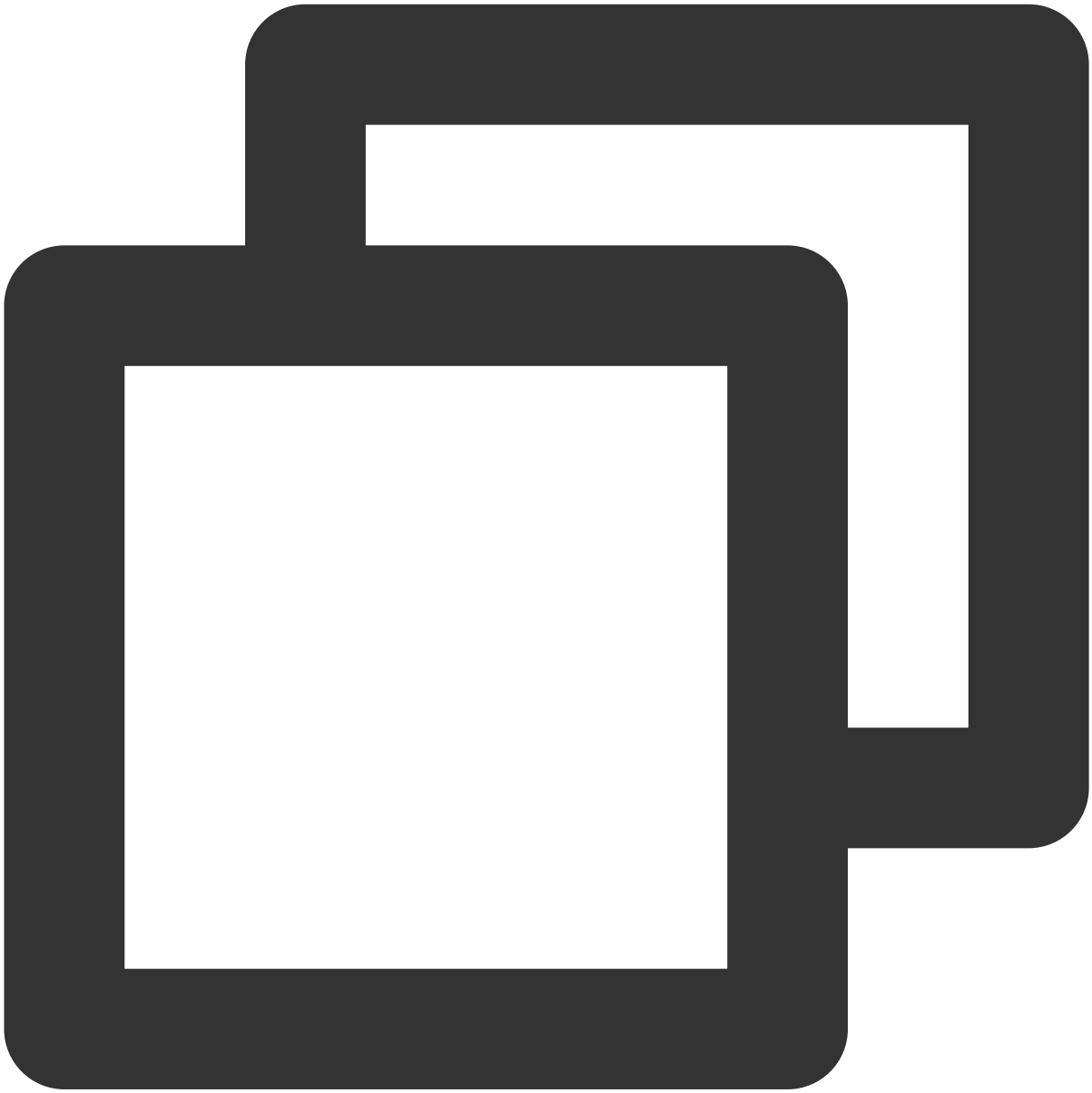
API Gateway provides sample code for Python v2.7 and v3. Please select one according to the version of your Python.

Notes

The eventually delivered HTTP request contains at least two headers: `Date` or `X-Date` and `Authorization`. More optional headers can be added in the request. If `Date` is used, the server will not check the time; if `X-Date` is used, the server will check the time.
The value of `Date` header is the construction time of the HTTP request in GMT format, such as Fri, 09 Oct 2015 00:00:00 GMT.
The value of `X-Date` header is the construction time of the HTTP request in GMT format, such as Mon, 19 Mar 2018 12:08:40 GMT. It cannot deviate from the current time for more than 15 minutes.

If it is a microservice API, you need to add two fields in the header: `X-NameSpace-Code` and `X-MicroService-Name`. They are not needed for general APIs and are included in the demo by default.

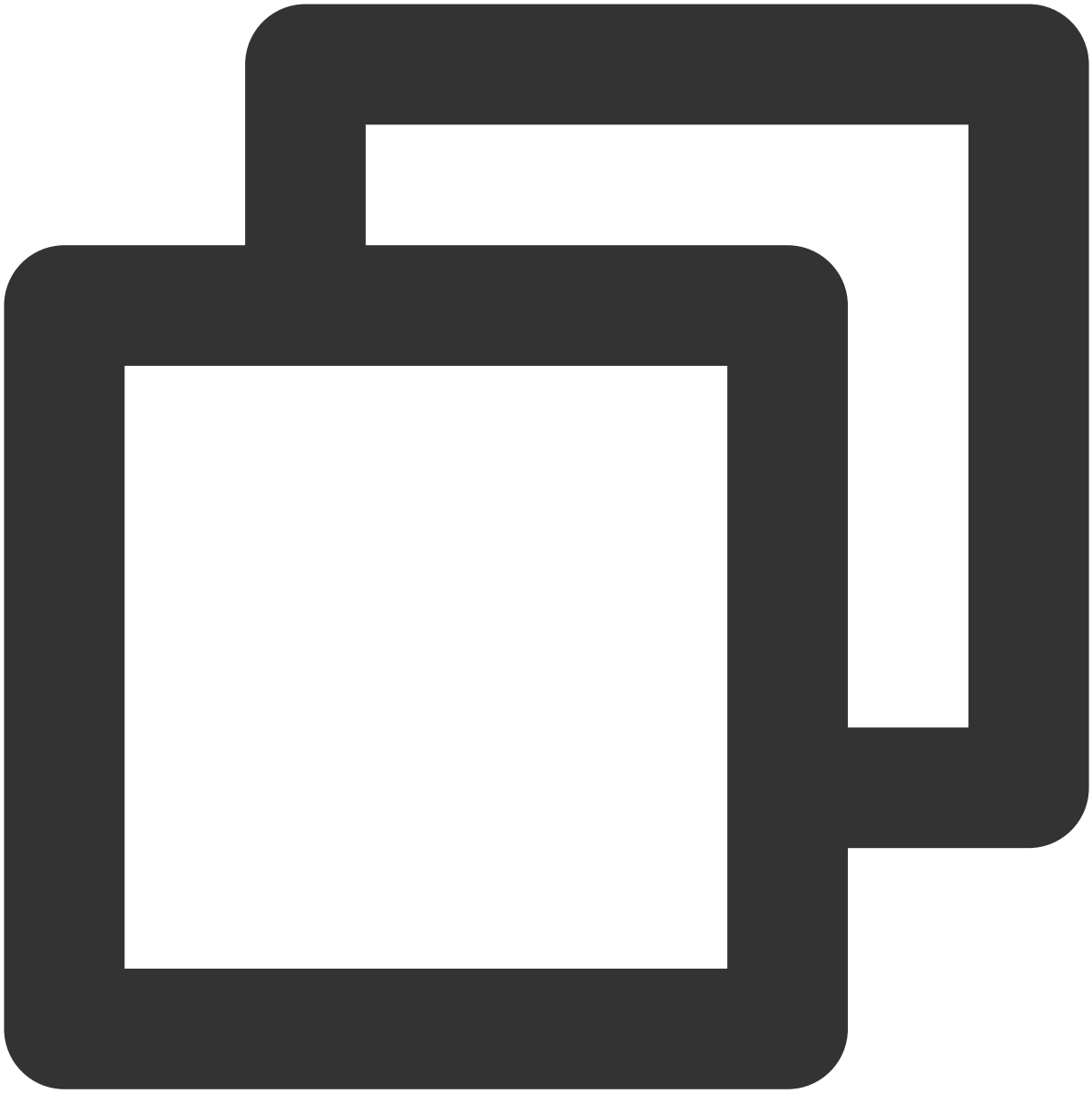
Sample code for Python v2.7



```
# -*- coding: utf-8 -*-
import requests
import datetime
import hashlib
from hashlib import sha1
import hmac
```

```
import base64


GMT_FORMAT = '%a, %d %b %Y %H:%M:%S GMT'

def getSimpleSign(source, SecretId, SecretKey) :
    dateTime = datetime.datetime.utcnow().strftime(GMT_FORMAT)
    auth = "hmac id=\\"" + SecretId + "\\", algorithm=\\"hmac-sha1\\", headers=\\"d
    signStr = "date: " + dateTime + "\\n" + "source: " + source
    sign = hmac.new(SecretKey, signStr, hashlib.sha1).digest()
    sign = base64.b64encode(sign)
    sign = auth + sign + "\\""
    return sign, dateTime

SecretId = 'your SecretId' # `SecretId` in key pair
SecretKey = 'your SecretKey' # `SecretKey` in key pair
url = 'http://service-xxxxxx-1234567890.ap-guangzhou.apigateway.myqcloud.com/releas

#header = {}
header = { 'Host':'service-xxxxxx-1234567890.ap-guangzhou.apigateway.myqcloud.com',
            'Accept': 'text/html, */*; q=0.01',
            'X-Requested-With': 'XMLHttpRequest',
            'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTM
            'Accept-Encoding': 'gzip, deflate, sdch',
            'Accept-Language': 'zh-CN,zh;q=0.8,ja;q=0.6'
}

Source = 'xxxxxx' # Arbitrary signature watermark value
sign, dateTime = getSimpleSign(Source, SecretId, SecretKey)
header['Authorization'] = sign
header['Date'] = dateTime
header['Source'] = Source

# If it is a microservice API, you need to add two fields in the header: 'X-NameSpa
header['X-NameSpace-Code'] = 'testmic'
header['X-MicroService-Name'] = 'provider-demo'

print header

r = requests.get(url, headers=header)
print r
print r.text
```

Sample code for Python v3



```
# -*- coding: utf-8 -*-
import base64
import datetime
import hashlib
import hmac

import requests

GMT_FORMAT = '%a, %d %b %Y %H:%M:%S GMT'
```

```
def getSimpleSign(source, SecretId, SecretKey):
    dateTime = datetime.datetime.utcnow().strftime(GMT_FORMAT)
    auth = "hmac id=\\"" + SecretId + "\\", algorithm=\\"hmac-sha1\\", headers=\\"d
    signStr = "date: " + dateTime + "\\n" + "source: " + source
    sign = hmac.new(SecretKey.encode(), signStr.encode(), hashlib.sha1).digest()
    sign = base64.b64encode(sign).decode()
    sign = auth + sign + "\\""
    return sign, dateTime


SecretId = 'your SecretId'  # SecretId in key pair
SecretKey = 'your SecretKey'  # SecretKey in key pair
url = 'http://service-xxxxxx-1234567890.ap-guangzhou.apigateway.myqcloud.com/releas

header = {'Host': 'service-xxxxxx-1234567890.ap-guangzhou.apigateway.myqcloud.com',
          'Accept': 'text/html, */*; q=0.01',
          'X-Requested-With': 'XMLHttpRequest',
          'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
          'Accept-Encoding': 'gzip, deflate, sdch',
          'Accept-Language': 'zh-CN,zh;q=0.8,ja;q=0.6'
          }
Source = 'xxxxxx'  # Arbitrary signature watermark value
sign, dateTime = getSimpleSign(Source, SecretId, SecretKey)
header['Authorization'] = sign
header['Date'] = dateTime
header['Source'] = Source


r = requests.get(url, headers=header)
print(r.text)
```

JavaScript (Key Pair Authentication)

Last updated：2023-12-22 10:08:49

Operation Scenarios

This document describes how to authenticate and manage your APIs through key pair authentication in JavaScript.

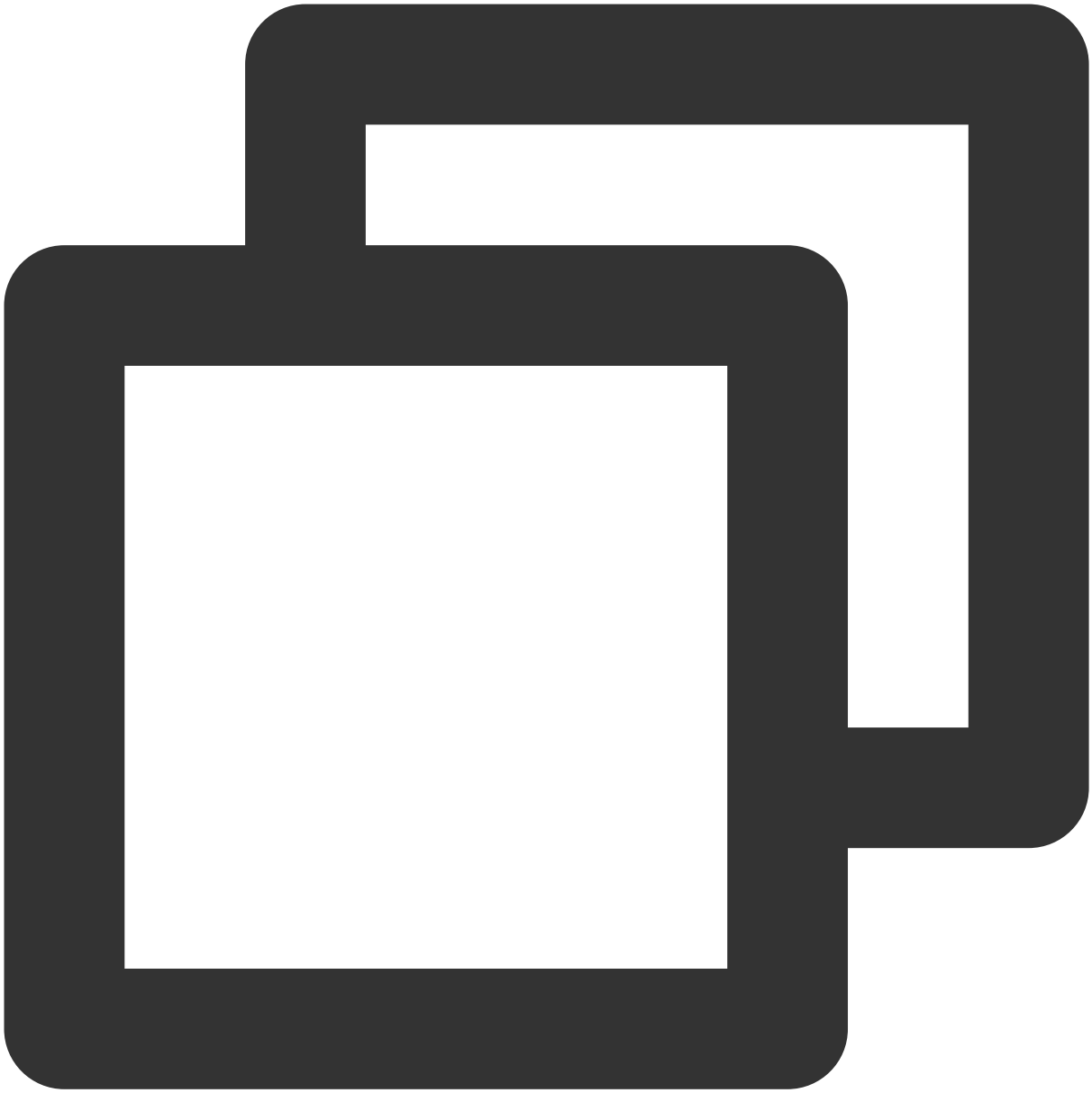
Directions

1. In the API Gateway Console, create an API and select the authentication type as "key pair authentication" (for more information, please see API Creation Overview).
2. Publish the service where the API resides to the release environment (for more information, please see Service Release and Deactivation).
3. Create a key pair on the key management page in the console.
4. Create a usage plan on the usage plan page in the console and bind it to the created key pair (for more information, please see Sample Usage Plan).
5. Bind the usage plan to the API or the service where the API resides.
6. Generate signing information in JavaScript by referring to the Sample Code.

Notes

The eventually delivered HTTP request contains at least two headers: `Date` or `X-Date` and `Authorization`. More optional headers can be added in the request. If `Date` is used, the server will not check the time; if `X-Date` is used, the server will check the time.
The value of `Date` header is the construction time of the HTTP request in GMT format, such as Fri, 09 Oct 2015 00:00:00 GMT.
The value of `X-Date` header is the construction time of the HTTP request in GMT format, such as Mon, 19 Mar 2018 12:08:40 GMT. It cannot deviate from the current time for more than 15 minutes.
If it is a microservice API, you need to add two fields in the header: `X-NameSpace-Code` and `X-MicroService-Name`. They are not needed for general APIs and are included in the demo by default.

Sample Code



```
/*
used...
<script src="js/vue.js"></script>
<script src="js/axios.js"></script>
<script src="js/qs.js"></script>
<script src="js/crypto-js/crypto-js.js"></script>
*/
var nowDate = new Date();

var dateTime = nowDate.toUTCString();
//dateTime = "Mon, 19 Mar 2018 12:00:44 GMT"
```

```
var SecretId = 'your SecretId'; // `SecretId` in key pair
var SecretKey = 'your SecretKey'; // `SecretKey` in key pair
var source = 'xxxxxx'; // Arbitrary signature watermark value
var auth = "hmac id=\\"" + SecretId + "\\", algorithm=\\"hmac-sha1\\", headers=\\"x
var signStr = "x-date: " + dateTime + "\\n" + "source: " + source;
console.log(signStr)
var sign = CryptoJS.HmacSHA1(signStr, SecretKey)
console.log(sign.toString())
sign = CryptoJS.enc.Base64.stringify(sign)
sign = auth + sign + "\\""
console.log(sign)
console.log(dateTime)

var instance = axios.create({
    baseURL: 'http://service-xxxxxxxx-1234567890.ap-guangzhou.apigateway.myqcloud.c
    timeout: 5000,
    headers: {
                "Source":source,
                "X-Date":dateTime,
                "Authorization":sign

                // If it is a microservice API, you need to add two fields in the h
                "X-NameSpace-Code": "testmic",
                "X-MicroService-Name": "provider-demo",
    },
    withCredentials: true
});

instance.get()
.then(function (response) {
        console.log(response);
    })
    .catch(function (error) {
        console.log(error);
    });;
```

PHP (Key Pair Authentication)

Last updated：2023-12-22 10:09:00

Operation Scenarios

This document describes how to authenticate and manage your APIs through key pair authentication in PHP.

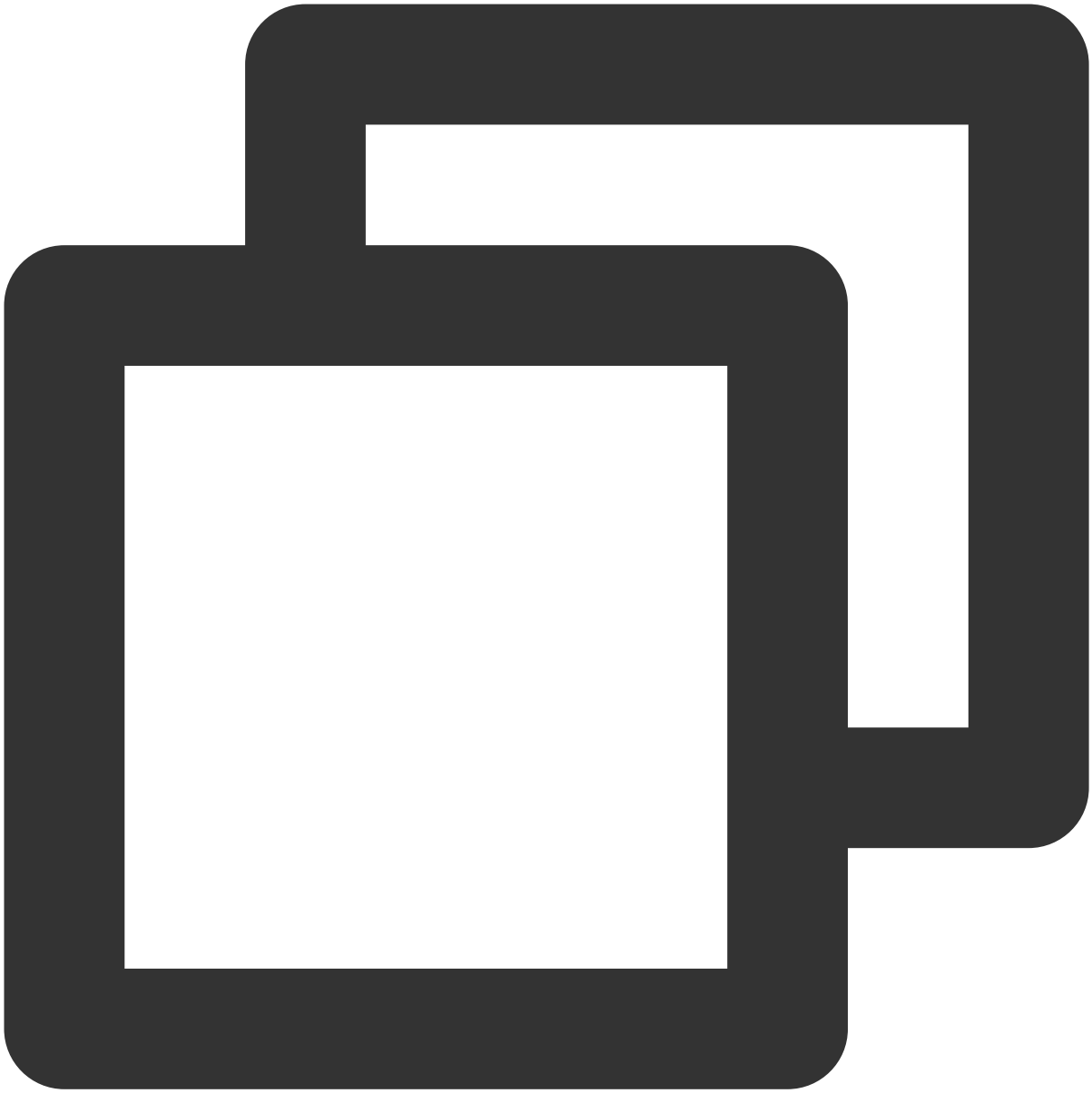
Directions

1. In the API Gateway Console, create an API and select the authentication type as "key pair authentication" (for more information, please see API Creation Overview).
2. Publish the service where the API resides to the release environment (for more information, please see Service Release and Deactivation).
3. Create a key pair on the key management page in the console.
4. Create a usage plan on the usage plan page in the console and bind it to the created key pair (for more information, please see Sample Usage Plan).
5. Bind the usage plan to the API or the service where the API resides.
6. Generate signing information in PHP by referring to the Sample Code.

Notes

The eventually delivered HTTP request contains at least two headers: `Date` or `X-Date` and `Authorization`. More optional headers can be added in the request. If `Date` is used, the server will not check the time; if `X-Date` is used, the server will check the time.
The value of `Date` header is the construction time of the HTTP request in GMT format, such as Fri, 09 Oct 2015 00:00:00 GMT.
The value of `X-Date` header is the construction time of the HTTP request in GMT format, such as Mon, 19 Mar 2018 12:08:40 GMT. It cannot deviate from the current time for more than 15 minutes.
If it is a microservice API, you need to add two fields in the header: `X-NameSpace-Code` and `X-MicroService-Name`. They are not needed for general APIs and are included in the demo by default.

Sample Code



```
<?php

$dateTime = gmdate("D, d M Y H:i:s T");
$SecretId = 'your SecretId'; # `SecretId` in key pair
$SecretKey = 'your SecretKey'; # `SecretKey` in key pair
$srcStr = "date: ".$dateTime."\\n"."source: "."xxxxxx"; # Arbitrary signature water
$Authen = 'hmac id="'.$SecretId.'", algorithm="hmac-sha1", headers="date source", s
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $SecretKey, true));
# echo $signStr;
$Authen = $Authen.$signStr."\\"";
echo $Authen;
```

```
# echo '</br>';

$url = 'http://service-xxxxxxxx-1234567890.ap-guangzhou.apigateway.myqcloud.com/rel
$headers = array(
    'Host:service-xxxxxxxx-1234567890.ap-guangzhou.apigateway.myqcloud.com', # Serv
    'Accept:text/html, */*; q=0.01',
    'Source: xxxxxx',
    'Date: '.$dateTime,
    'Authorization: '.$Authen,
    'X-Requested-With: XMLHttpRequest',
    # 'Accept-Encoding: gzip, deflate, sdch',

    # If it is a microservice API, you need to add two fields in the header: 'X-Nam
    'X-NameSpace-Code: testmic',
    'X-MicroService-Name: provider-demo'
);
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL,$url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_TIMEOUT, 60);
curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "GET");

$data = curl_exec($ch);

if (curl_errno($ch)) {
    print "Error: " . curl_error($ch);
} else {
    # Show me the result
    var_dump($data);
    curl_close($ch);
}

?>
```

C++ (Key Pair Authentication)

Last updated：2023-12-22 10:09:13

Operation Scenarios

This document describes how to authenticate and manage your APIs through key pair authentication in C++.

Directions

1. In the API Gateway Console, create an API and select the authentication type as "key pair" (for more information, please see API Creation Overview).
2. Publish the service where the API resides to the release environment (for more information, please see Service Release and Deactivation).
3. Create a key pair on the key management page in the console.
4. Create a usage plan on the usage plan page in the console and bind it to the created key pair (for more information, please see Sample Usage Plan).
5. Bind the usage plan to the API or the service where the API resides.
6. Generate signing information in C++ by referring to the Sample Code.

Environment Dependencies

In this demo, libcurl is used to initiate HTTP requests, so the compiler machine needs to have the libcurl library installed.

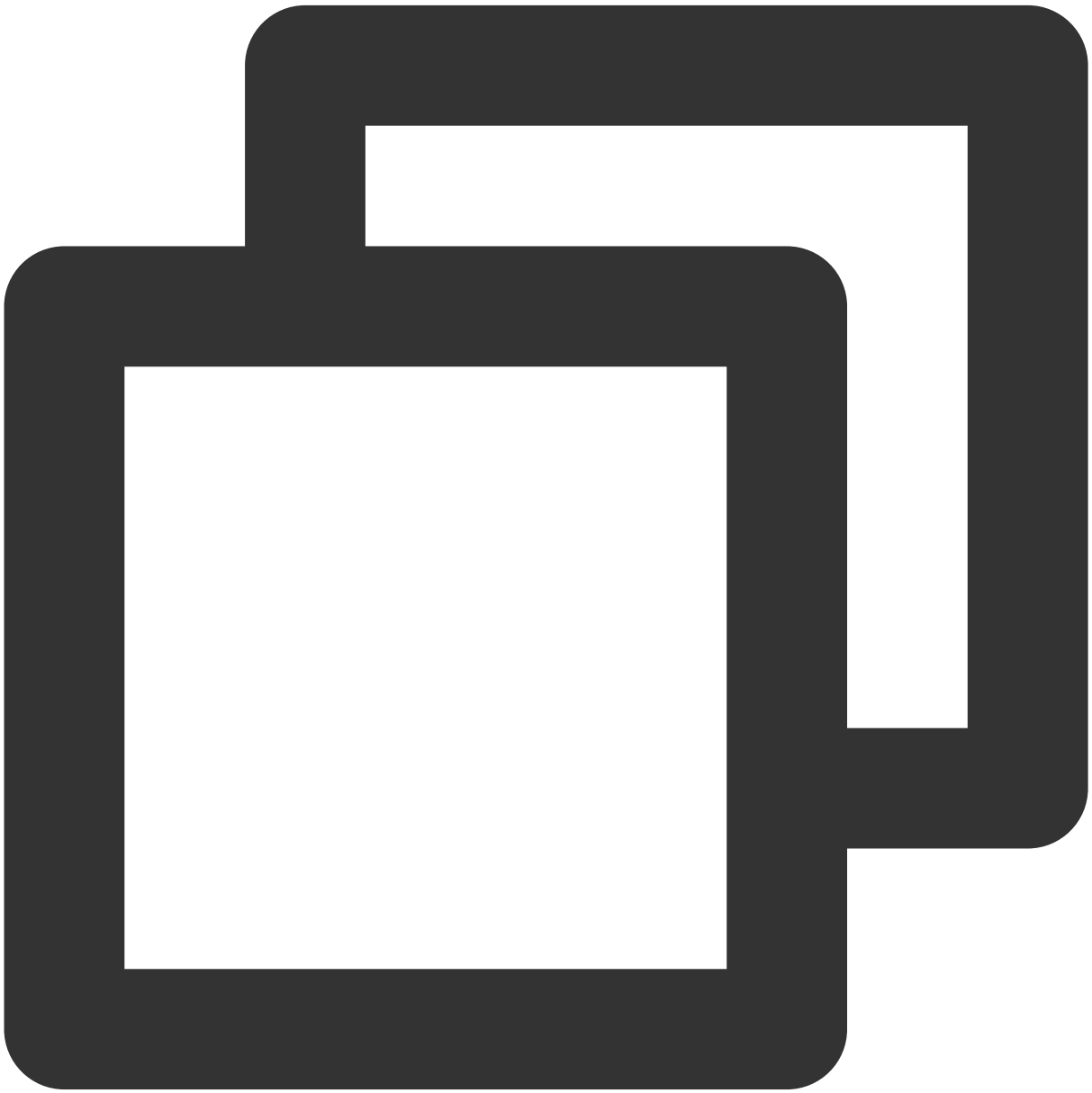
Notes

The eventually delivered HTTP request contains at least two headers: `Date` or `X-Date` and `Authorization`. More optional headers can be added in the request. If `Date` is used, the server will not check the time; if `X-Date` is used, the server will check the time.
The value of `Date` header is the construction time of the HTTP request in GMT format, such as Fri, 09 Oct 2015 00:00:00 GMT.
The value of `X-Date` header is the construction time of the HTTP request in GMT format, such as Mon, 19 Mar 2018 12:08:40 GMT. It cannot deviate from the current time for more than 15 minutes.

If it is a microservice API, you need to add two fields in the header: `X-NameSpace-Code` and `X-MicroService-Name`. They are not needed for general APIs and are included in the demo by default.

Directory Structure

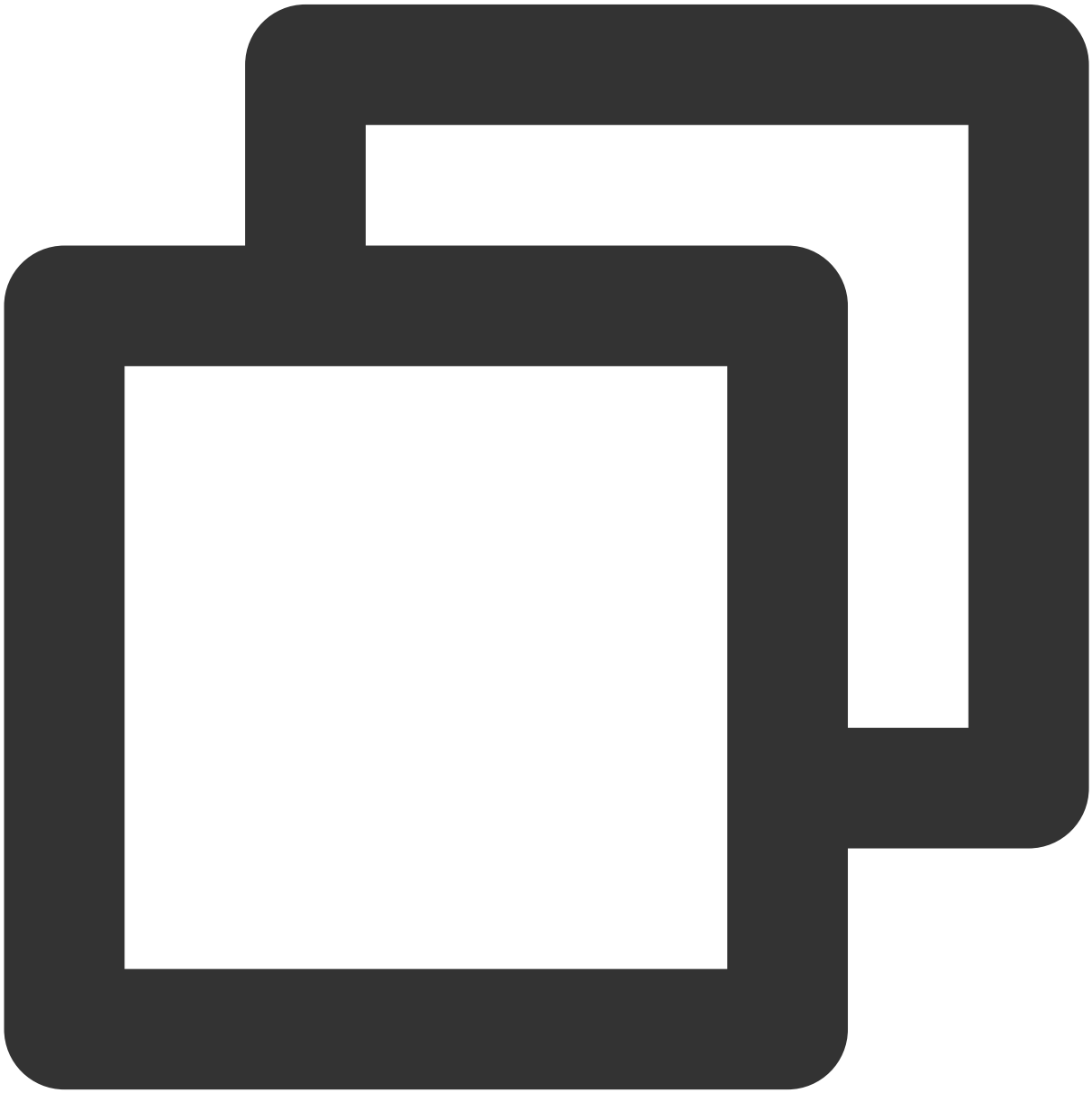
This demo contains 7 files in total, and the directory structure is as follows:



```
├─AuthenticationDemo.cpp
├─request.cpp
```

```
├─base64.h
├─base64.cpp
├─hmac.h
├─sha1.h
└─sha1.cpp
```

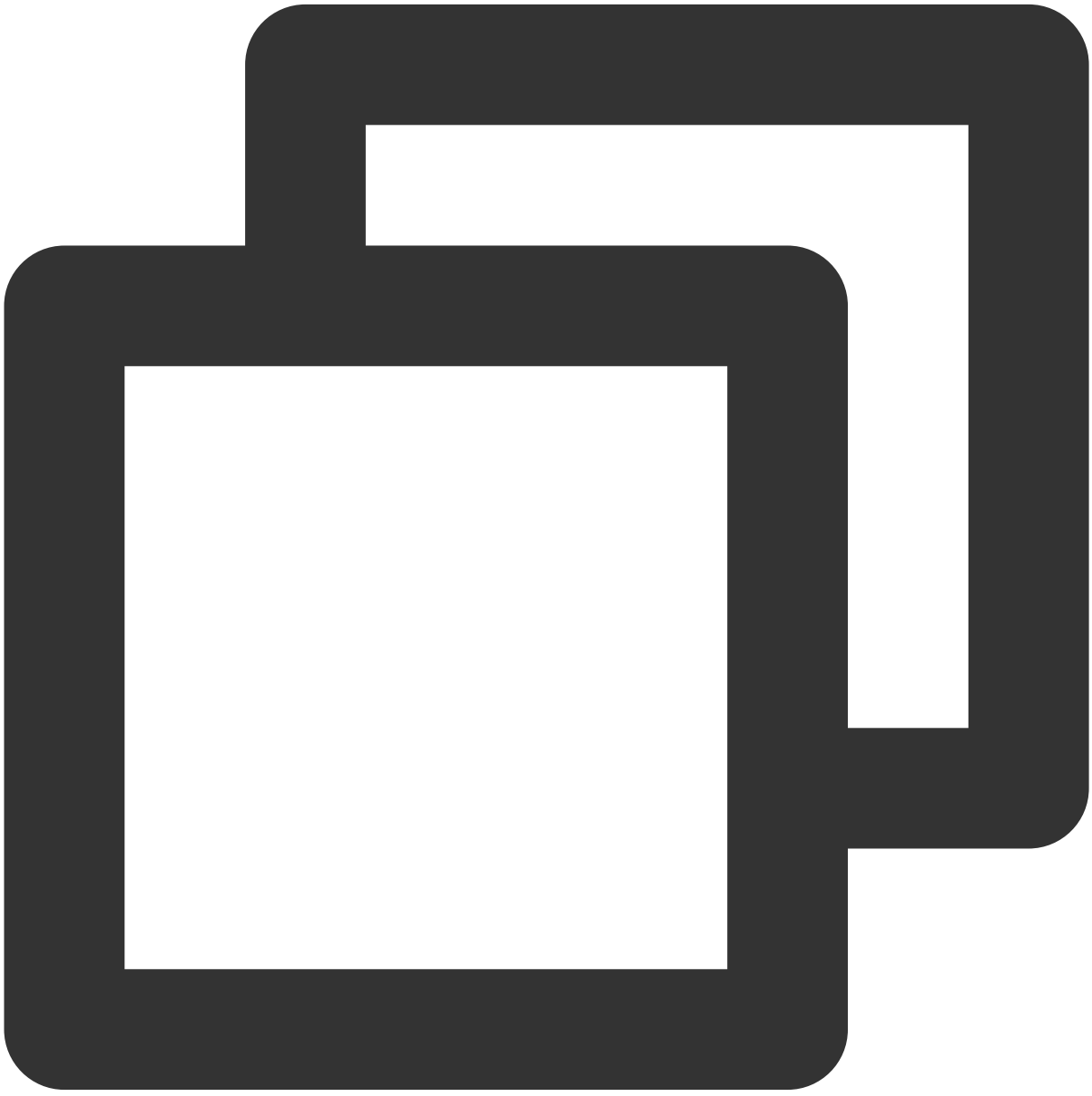
Compilation Command



```
g++ -o AuthenticationDemo AuthenticationDemo.cpp request.cpp base64.cpp sha1.cpp -l
```

Sample Code

AuthenticationDemo.cpp

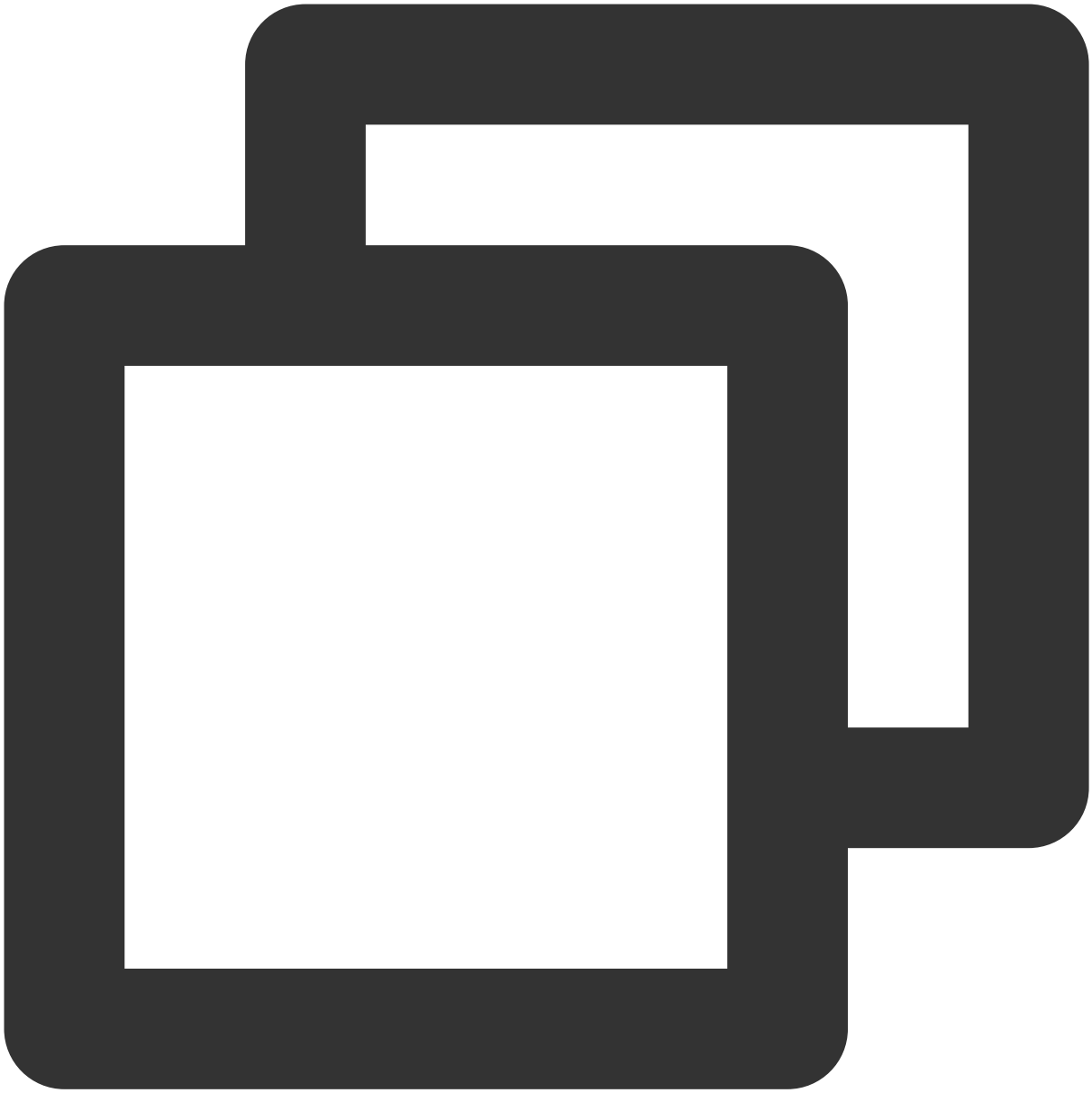


```
/*In this Demo, libcurl is used to initiate HTTP requests, so the compiler machine
/*Compilation command: g++ -o AuthenticationDemo AuthenticationDemo.cpp request.cpp
```

```
#include <iostream>
#include <stdio.h>
#include"hmac.h"
#include"sha1.h"
#include"base64.h"

extern void get_request(const string &defaultDomain, const string &source, const st
extern void post_request(const string &defaultDomain, const string &source, const s

using namespace std;

void GetGmtTime(string &szGmtTime)
{
    time_t rawTime;
    struct tm* timeInfo;
    char szTemp[30]={0};
    time(&rawTime);
    timeInfo = gmtime(&rawTime);
    strftime(szTemp,sizeof(szTemp),"%a, %d %b %Y %H:%M:%S GMT",timeInfo);
    szGmtTime = szTemp;
}

int calcAuthorization(const string &source, const string &secretId, const string &s
{
    GetGmtTime(dateTime);
    sign = "x-date: " + dateTime + "\\nsource: " + source;
    sign = hmac<SHA1>(sign, secretKey);
    string binDight;
    HexToBin(sign , binDight);
    BinToBase64(binDight , sign);
    char tempauth[1024] = {0};
    snprintf(tempauth,sizeof(tempauth)-1,"hmac id=\\"%s\\", algorithm=\\"hmac-sha1\
    sign = tempauth;

    return 0;
}


/*Enter the `secretId`, `secretKey`, `defaultDomain`, and `reqUrl` in the code belo

int main()
{
    const string secretId = "your secretId";// `SecretId` in key pair
    const string secretKey = "your secretKey";// `SecretKey` in key pair
    const string source = "xxxxxx"; // Arbitrary signature watermark value
    string sign, dateTime;
```

```
    calcAuthorization(source, secretId, secretKey, sign, dateTime);
    const string defaultDomain = "service-xxxxxxxx-1234567890.ap-guangzhou.apigatew
    const string reqUrl = "https://service-xxxxxxxx-1234567890.ap-guangzhou.apigate

    get_request(defaultDomain, source, dateTime, sign, reqUrl);
    //post_request(defaultDomain, source, dateTime, sign, reqUrl);

    return 0;
}
```

request.cpp



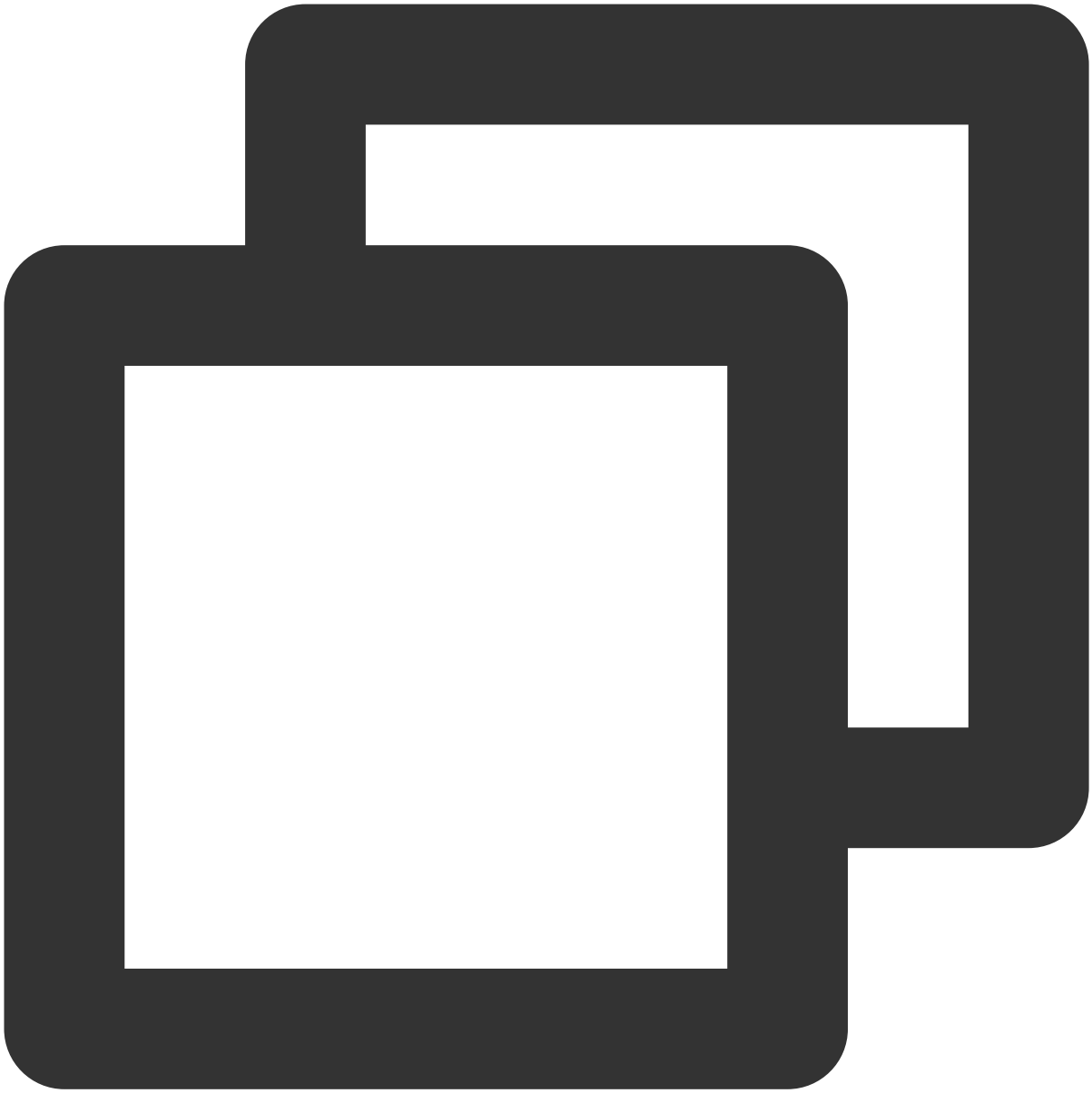
```
#include <iostream>
#include <cstring>
#include "curl/curl.h"

using namespace std;

size_t req_reply(void *ptr, size_t size, size_t nmemb, void *stream)
{
    ((std::string*)stream)->append((char*)ptr, size*nmemb);
    return size * nmemb;
}
```

```
void get_request(const string &defaultDomain, const string &source, const string &d
{
    CURL* curl = curl_easy_init();
    if (curl)
    {
        curl_easy_setopt(curl, CURLOPT_URL, reqUrl.c_str());
        curl_easy_setopt(curl, CURLOPT_SSL_VERIFYPEER, 0L);
        struct curl_slist * slist = NULL;
        slist = curl_slist_append(slist, "Accept:*/*");
        slist = curl_slist_append(slist, "Accept-Charset:utf-8;");
        string headDomain = "Host:" + defaultDomain;
        slist = curl_slist_append(slist, headDomain.c_str());
        string headSource = "Source:" + source;
        slist = curl_slist_append(slist, headSource.c_str());
        string headDatetime = "X-Date:" + dateTime;
        slist = curl_slist_append(slist, headDatetime.c_str());
        string headAuthorization = "Authorization:" + sign;
        slist = curl_slist_append(slist, headAuthorization.c_str());
        // If it is a microservice API, you need to add two fields in the header: '
        slist = curl_slist_append(slist, "x-NameSpace-Code:testmic");
        slist = curl_slist_append(slist, "x-MicroService-Name:provider-demo");
        curl_easy_setopt(curl, CURLOPT_HTTPHEADER, slist);
        curl_easy_setopt(curl, CURLOPT_CONNECTTIMEOUT, 5);
        curl_easy_setopt(curl, CURLOPT_TIMEOUT, 5);
        curl_easy_setopt(curl, CURLOPT_READFUNCTION, NULL);
        std::string response_data;
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, &response_data);
        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, req_reply);

        CURLcode res = curl_easy_perform(curl);
        if (res != CURLE_OK)
        {
            fprintf(stderr, "curl_easy_perform() failed: %s\\n",
            curl_easy_strerror(res));
        }
        else
        {
            // get response code
            long response_code;
            curl_easy_getinfo(curl, CURLINFO_RESPONSE_CODE, &response_code);
            printf("response code %d \\n", response_code);
            printf("response data : %s\\n ",response_data.c_str());
        }
        curl_slist_free_all(slist);
        curl_easy_cleanup(curl);
    }
```

```
    curl_global_cleanup();
}

void post_request(const string &defaultDomain, const string &source, const string &
{
    CURL* curl = curl_easy_init();
    if (curl)
    {
        curl_easy_setopt(curl, CURLOPT_URL, reqUrl.c_str());
        curl_easy_setopt(curl, CURLOPT_SSL_VERIFYPEER, 0L);
        curl_easy_setopt(curl, CURLOPT_POST, 1);
        struct curl_slist * slist = NULL;
        slist = curl_slist_append(slist, "Accept:*/*");
        slist = curl_slist_append(slist, "Accept-Charset:utf-8;");
        string headDomain = "Host:" + defaultDomain;
        slist = curl_slist_append(slist, headDomain.c_str());
        string headSource = "Source:" + source;
        slist = curl_slist_append(slist, headSource.c_str());
        string headDatetime = "X-Date:" + dateTime;
        slist = curl_slist_append(slist, headDatetime.c_str());
        string headAuthorization = "Authorization:" + sign;
        slist = curl_slist_append(slist, headAuthorization.c_str());
        // If it is a microservice API, you need to add two fields in the header: '
        slist = curl_slist_append(slist, "x-NameSpace-Code:testmic");
        slist = curl_slist_append(slist, "x-MicroService-Name:provider-demo");
        curl_easy_setopt(curl, CURLOPT_HTTPHEADER, slist);

         // set body
        std::string body = "{\\
            \\"title\\":\\"post title\\",\\
            \\"body\\" : \\"post body\\",\\
            \\"userId\\" : 1}";
        curl_easy_setopt(curl, CURLOPT_POSTFIELDS, body.c_str());

        curl_easy_setopt(curl, CURLOPT_CONNECTTIMEOUT, 5);
        curl_easy_setopt(curl, CURLOPT_TIMEOUT, 5);
        curl_easy_setopt(curl, CURLOPT_READFUNCTION, NULL);
        std::string response_data;
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, &response_data);
        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, req_reply);

        CURLcode res = curl_easy_perform(curl);
        if (res != CURLE_OK)
        {
            fprintf(stderr, "curl_easy_perform() failed: %s\\n",
            curl_easy_strerror(res));
        }
```

```
        else
        {
            // get response code
            long response_code;
            curl_easy_getinfo(curl, CURLINFO_RESPONSE_CODE, &response_code);
            printf("response code %d \\n", response_code);
            printf("response data : %s\\n ",response_data.c_str());
        }
        curl_slist_free_all(slist);
        curl_easy_cleanup(curl);
    }
    curl_global_cleanup();
}
```

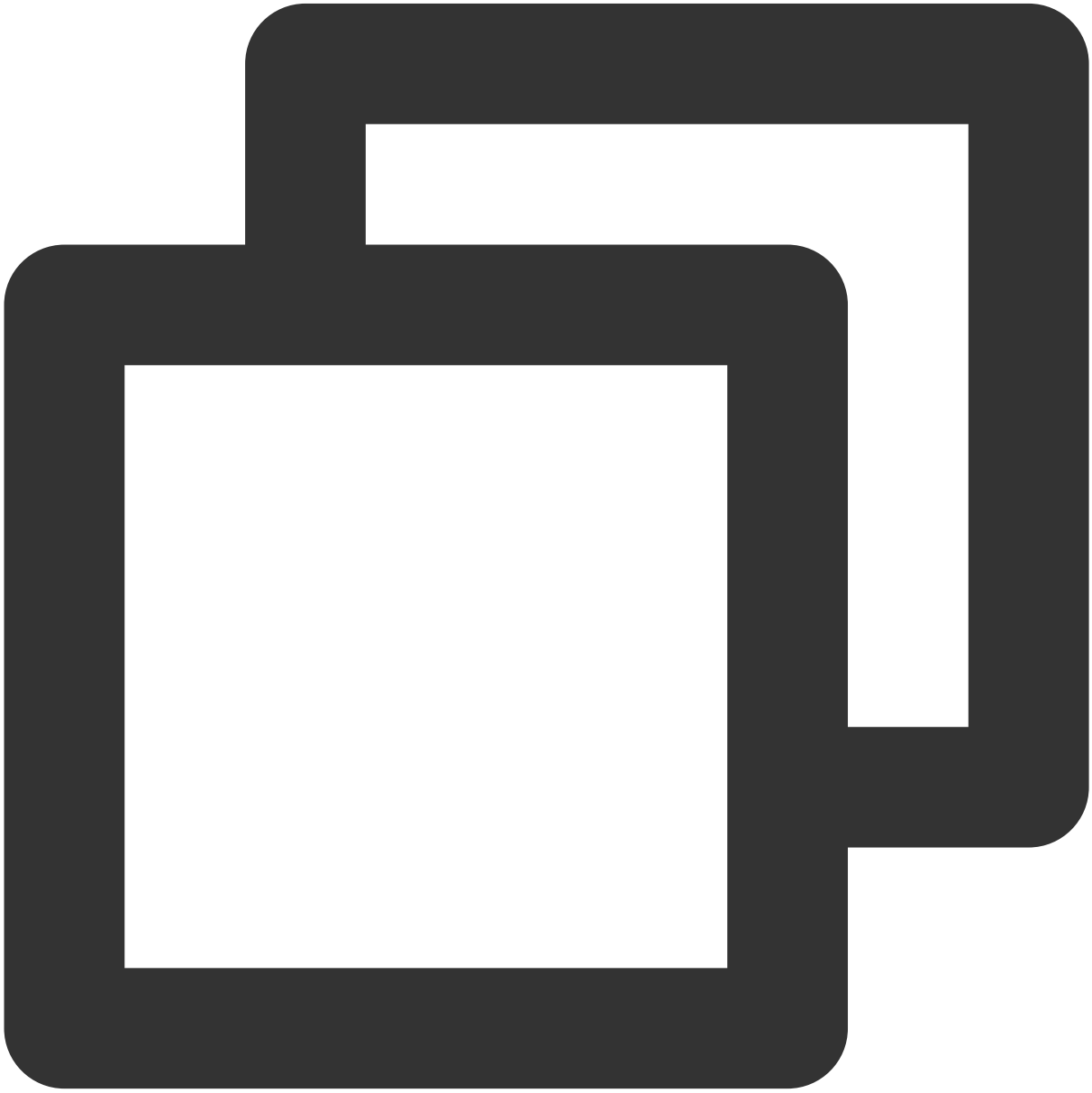
base64.h



```
// Base64 encoding table
#include<string>
using namespace std;
const  char Base64EncodeMap[64] =
{
    'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H',
    'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',
    'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X',
    'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f',
    'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n',
    'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
```

```
    'w', 'x', 'y', 'z', '0', '1', '2', '3',
    '4', '5', '6', '7', '8', '9', '+', '/'
};

int BinToDecInt(string strBin);
void BinToBase64(string binStr , string &base64Str);
void HexToBin(string hexDight , string& binDight);
```

base64.cpp



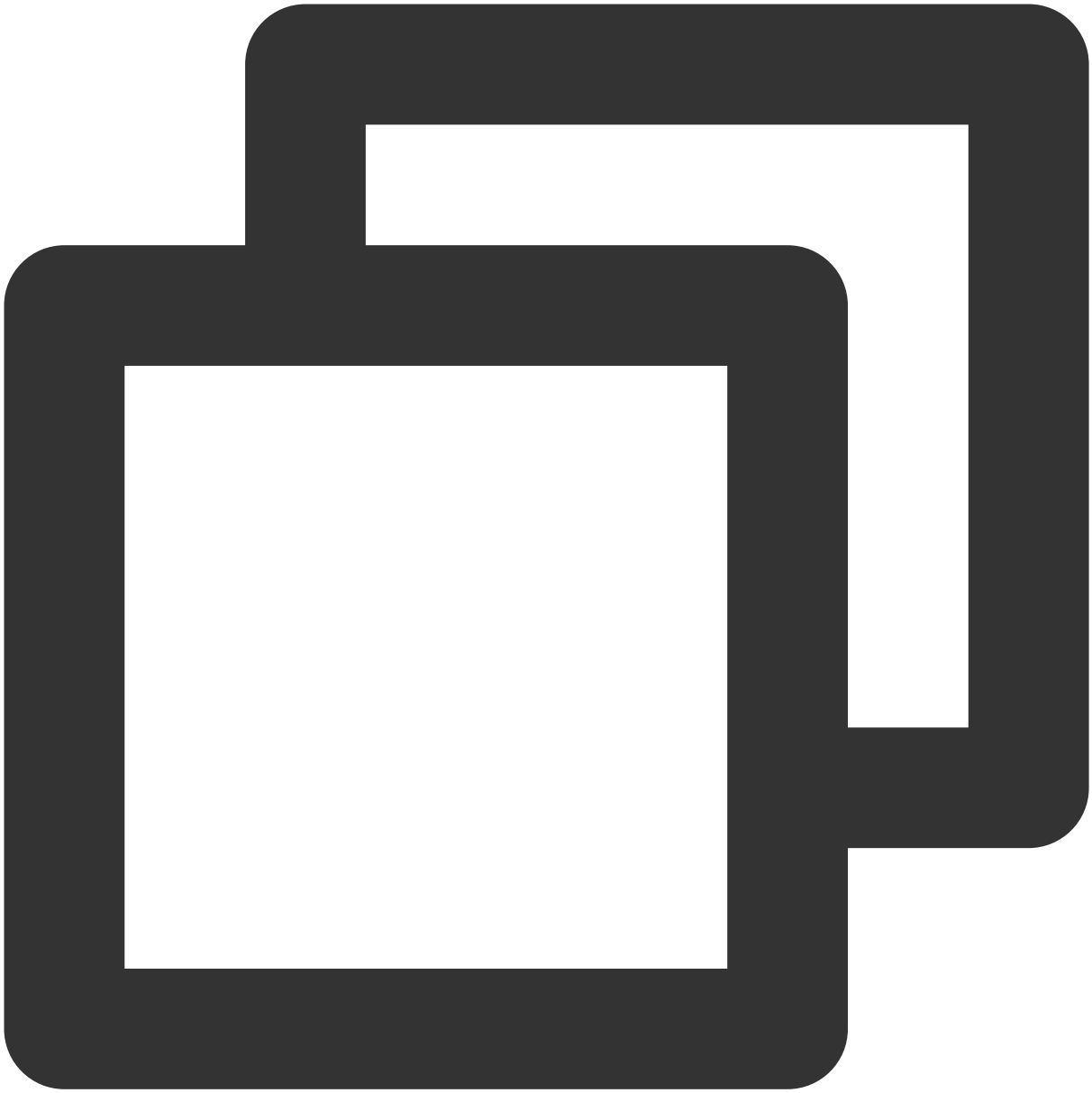
```
#include"base64.h"
```

```
int BinToDecInt(string strBin){
              int num = 0;
              int b = 0;
              for(int i = 0; i < strBin.length() ;i++){
                  num = num * 2;
                  b = static_cast<int>(strBin[i]-'0');
                  num = num + b;
              }
              return num;
        }

void BinToBase64(string binStr , string &base64Str)
{
    while(binStr.length() % 6 != 0){
        binStr = binStr + "0";
    }
    base64Str = "";
    string tmp = "";
    int index = 0;
    int num = 0;
    while(index < binStr.length()){
        tmp = binStr.substr(index , 6);
        index = index + 6;
        num = BinToDecInt(tmp);
        base64Str = base64Str + Base64EncodeMap[num];
    }
    base64Str = base64Str + "=";
}



void HexToBin(string hexDight , string& binDight){
    binDight = "";
    int f = 0,c = 0;
    char e;
    for(int f = 0; f < hexDight.length() ; f++){
        e = hexDight[f];
        if(e >= 'a' && e <= 'f'){
            int a = static_cast<int>(e-'a'+10);
            switch(a){
                case 10 : binDight = binDight + "1010";
                    break;
                case 11 : binDight = binDight + "1011";
                    break;
                case 12 : binDight = binDight + "1100";
                    break;
```

```
                case 13 : binDight = binDight + "1101";
                    break;
                case 14 : binDight = binDight + "1110";
                    break;
                case 15 : binDight = binDight + "1111";
                    break;
            }
        }
        else if( e >= '0' && e <= '9'){
            int b = static_cast<int>(e-'0');
            if(f == 0){
            switch(b){
                case 0: binDight = binDight + "0000";
                    break;
                case 1: binDight = binDight + "0001";
                    break;
                case 2: binDight = binDight + "0010";
                    break;
                case 3: binDight = binDight + "0011";
                    break;
                case 4: binDight = binDight + "0100";
                    break;
                case 5: binDight = binDight + "0101";
                    break;
                case 6: binDight = binDight + "0110";
                    break;
                case 7: binDight = binDight + "0111";
                    break;
                case 8: binDight = binDight + "1000";
                    break;
                case 9: binDight = binDight + "1001";
                    break;
            }
            }
            else{
                switch(b){
                case 0 : binDight = binDight + "0000";
                         break;
                case 1: binDight = binDight + "0001";
                    break;
                case 2: binDight = binDight + "0010";
                    break;
                case 3: binDight = binDight + "0011";
                    break;
                case 4: binDight = binDight + "0100";
                    break;
                case 5: binDight = binDight + "0101";
```

```
                    break;
                case 6: binDight = binDight + "0110";
                    break;
                case 7: binDight = binDight + "0111";
                    break;
                case 8: binDight = binDight + "1000";
                    break;
                case 9: binDight = binDight + "1001";
                    break;
                }
            }
        }
    }
}
```

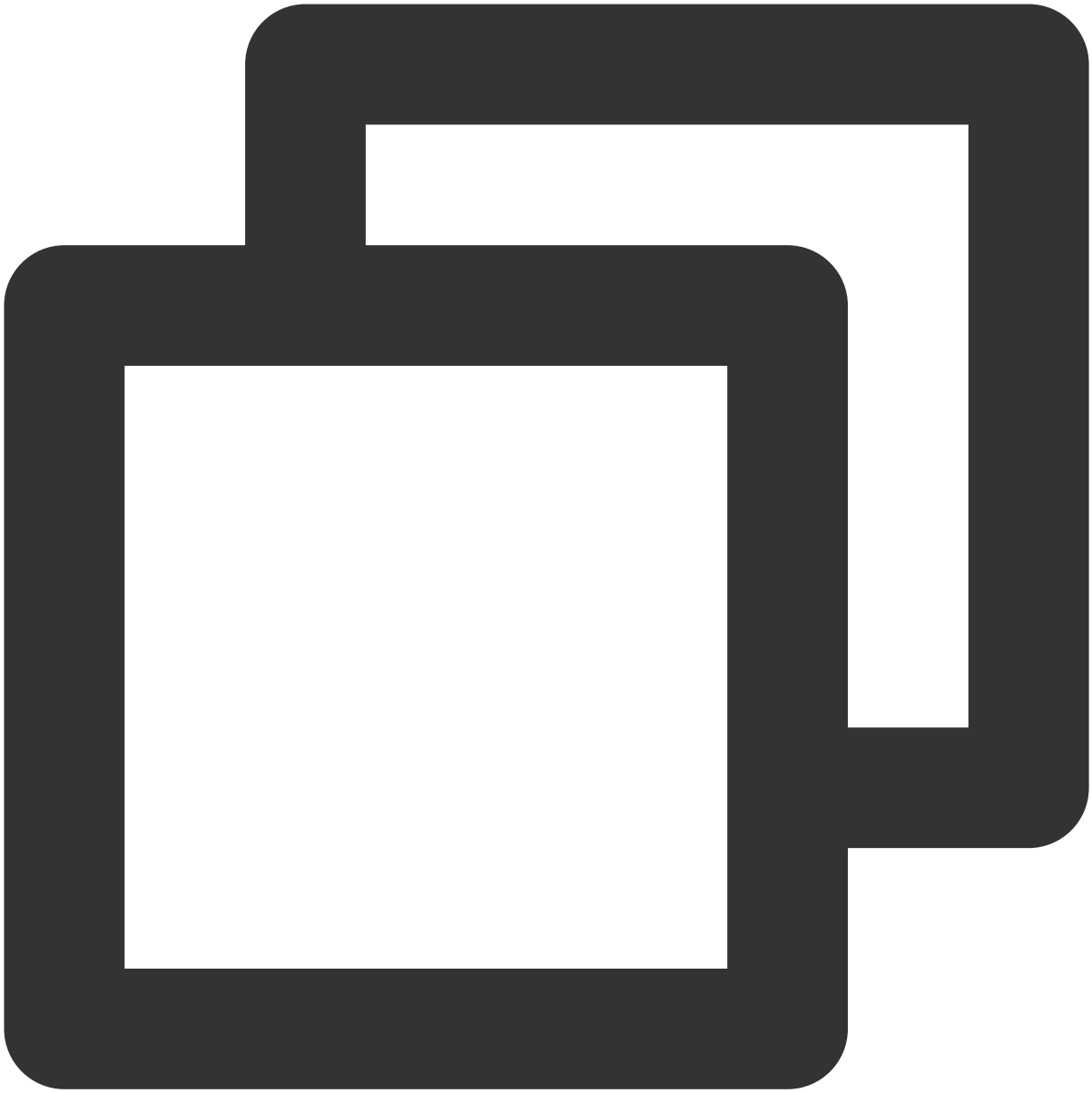
hmac.h



```
#pragma once


#include <string>
#include <cstring>

/// compute HMAC hash of data and key using MD5, SHA1 or SHA256
template <typename HashMethod>
std::string hmac(const void* data, size_t numDataBytes, const void* key, size_t num
{
  unsigned char usedKey[HashMethod::BlockSize] = {0};
```

```
  if (numKeyBytes <= HashMethod::BlockSize)
  {
    memcpy(usedKey, key, numKeyBytes);
  }
  else
  {
    HashMethod keyHasher;
    keyHasher.add(key, numKeyBytes);
    keyHasher.getHash(usedKey);
  }

  for (size_t i = 0; i < HashMethod::BlockSize; i++)
    usedKey[i] ^= 0x36;

  unsigned char inside[HashMethod::HashBytes];
  HashMethod insideHasher;
  insideHasher.add(usedKey, HashMethod::BlockSize);
  insideHasher.add(data,    numDataBytes);
  insideHasher.getHash(inside);

  for (size_t i = 0; i < HashMethod::BlockSize; i++)
    usedKey[i] ^= 0x5C ^ 0x36;

  HashMethod finalHasher;
  finalHasher.add(usedKey, HashMethod::BlockSize);
  finalHasher.add(inside,  HashMethod::HashBytes);

  return finalHasher.getHash();
}

template <typename HashMethod>
std::string hmac(const std::string& data, const std::string& key)
{
  return hmac<HashMethod>(data.c_str(), data.size(), key.c_str(), key.size());
}
```

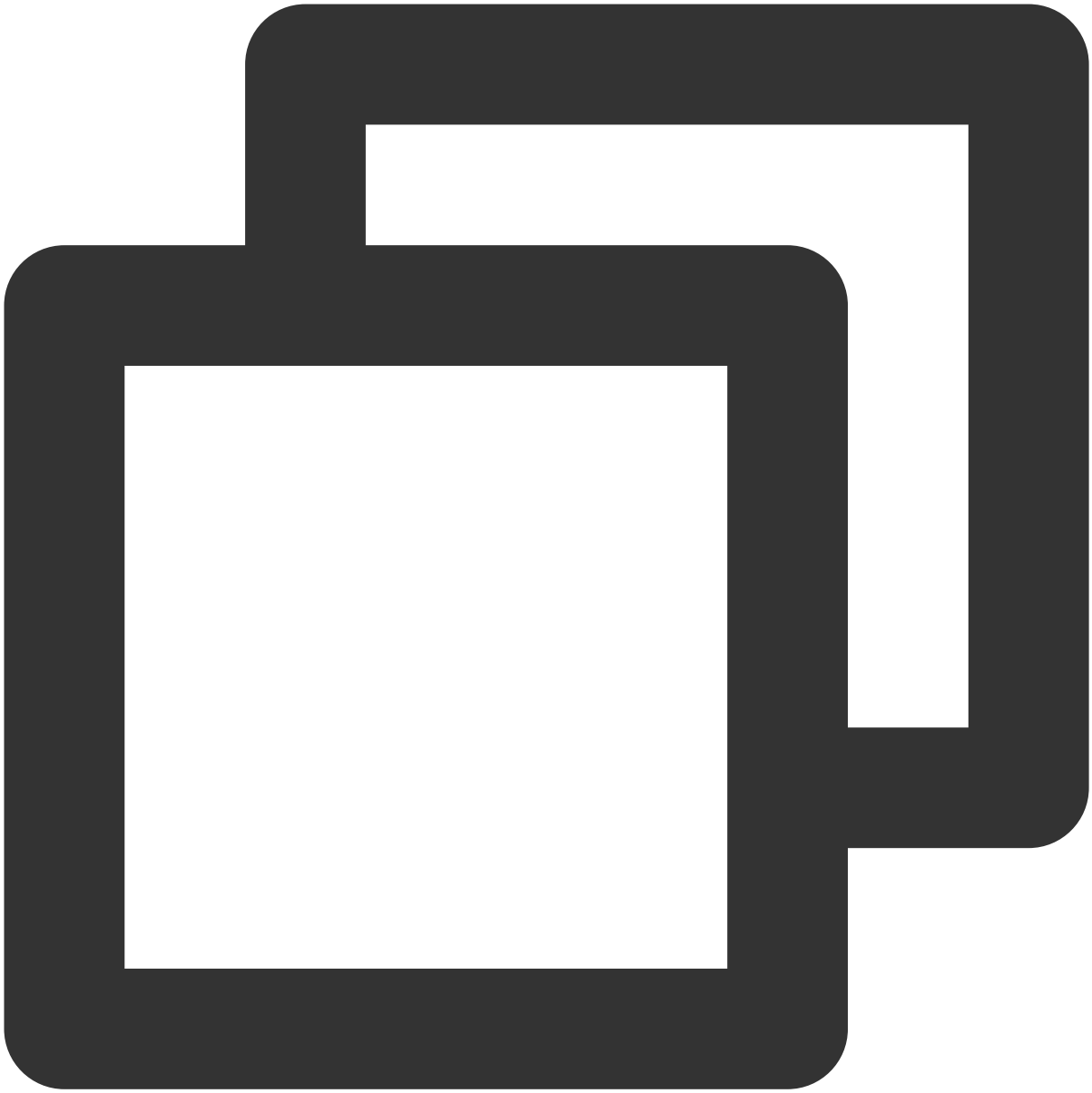
sha1.h



```
#pragma once

#include <string>

#ifdef _MSC_VER
// Windows
typedef unsigned __int8  uint8_t;
typedef unsigned __int32 uint32_t;
typedef unsigned __int64 uint64_t;
#else
// GCC
```

```
#include <stdint.h>
#endif


class SHA1 //: public Hash
{
public:
  enum { BlockSize = 512 / 8, HashBytes = 20 };

  SHA1();

  std::string operator()(const void* data, size_t numBytes);

  std::string operator()(const std::string& text);

  void add(const void* data, size_t numBytes);

  std::string getHash();

  void getHash(unsigned char buffer[HashBytes]);

  void reset();

private:
  void processBlock(const void* data);

  void processBuffer();

  uint64_t m_numBytes;

  size_t   m_bufferSize;

  uint8_t  m_buffer[BlockSize];

  enum { HashValues = HashBytes / 4 };

  uint32_t m_hash[HashValues];
};
```

sha1.cpp



```
#include "sha1.h"

#ifndef _MSC_VER
#include <endian.h>
#endif


SHA1::SHA1()
{
  reset();
}
```

```
void SHA1::reset()
{
  m_numBytes   = 0;
  m_bufferSize = 0;

  m_hash[0] = 0x67452301;
  m_hash[1] = 0xefcdab89;
  m_hash[2] = 0x98badcfe;
  m_hash[3] = 0x10325476;
  m_hash[4] = 0xc3d2e1f0;
}


namespace
{
  inline uint32_t f1(uint32_t b, uint32_t c, uint32_t d)
  {
    return d ^ (b & (c ^ d));
  }

  inline uint32_t f2(uint32_t b, uint32_t c, uint32_t d)
  {
    return b ^ c ^ d;
  }

  inline uint32_t f3(uint32_t b, uint32_t c, uint32_t d)
  {
    return (b & c) | (b & d) | (c & d);
  }

  inline uint32_t rotate(uint32_t a, uint32_t c)
  {
    return (a << c) | (a >> (32 - c));
  }

  inline uint32_t swap(uint32_t x)
  {
#if defined(__GNUC__) || defined(__clang__)
    return __builtin_bswap32(x);
#endif
#ifdef MSC_VER
    return _byteswap_ulong(x);
#endif

    return (x >> 24) |
```

```
            ((x >>  8) & 0x0000FF00) |
            ((x <<  8) & 0x00FF0000) |
             (x << 24);
  }
}

void SHA1::processBlock(const void* data)
{
  uint32_t a = m_hash[0];
  uint32_t b = m_hash[1];
  uint32_t c = m_hash[2];
  uint32_t d = m_hash[3];
  uint32_t e = m_hash[4];

  const uint32_t* input = (uint32_t*) data;

  uint32_t words[80];
  for (int i = 0; i < 16; i++)
#if defined(__BYTE_ORDER) && (__BYTE_ORDER != 0) && (__BYTE_ORDER == __BIG_ENDIAN)
    words[i] = input[i];
#else
    words[i] = swap(input[i]);
#endif

  for (int i = 16; i < 80; i++)
    words[i] = rotate(words[i-3] ^ words[i-8] ^ words[i-14] ^ words[i-16], 1);

  for (int i = 0; i < 4; i++)
  {
    int offset = 5*i;
    e += rotate(a,5) + f1(b,c,d) + words[offset  ] + 0x5a827999; b = rotate(b,30);
    d += rotate(e,5) + f1(a,b,c) + words[offset+1] + 0x5a827999; a = rotate(a,30);
    c += rotate(d,5) + f1(e,a,b) + words[offset+2] + 0x5a827999; e = rotate(e,30);
    b += rotate(c,5) + f1(d,e,a) + words[offset+3] + 0x5a827999; d = rotate(d,30);
    a += rotate(b,5) + f1(c,d,e) + words[offset+4] + 0x5a827999; c = rotate(c,30);
  }

  for (int i = 4; i < 8; i++)
  {
    int offset = 5*i;
    e += rotate(a,5) + f2(b,c,d) + words[offset  ] + 0x6ed9eba1; b = rotate(b,30);
    d += rotate(e,5) + f2(a,b,c) + words[offset+1] + 0x6ed9eba1; a = rotate(a,30);
    c += rotate(d,5) + f2(e,a,b) + words[offset+2] + 0x6ed9eba1; e = rotate(e,30);
    b += rotate(c,5) + f2(d,e,a) + words[offset+3] + 0x6ed9eba1; d = rotate(d,30);
    a += rotate(b,5) + f2(c,d,e) + words[offset+4] + 0x6ed9eba1; c = rotate(c,30);
  }
```

```
  for (int i = 8; i < 12; i++)
  {
    int offset = 5*i;
    e += rotate(a,5) + f3(b,c,d) + words[offset  ] + 0x8f1bbcdc; b = rotate(b,30);
    d += rotate(e,5) + f3(a,b,c) + words[offset+1] + 0x8f1bbcdc; a = rotate(a,30);
    c += rotate(d,5) + f3(e,a,b) + words[offset+2] + 0x8f1bbcdc; e = rotate(e,30);
    b += rotate(c,5) + f3(d,e,a) + words[offset+3] + 0x8f1bbcdc; d = rotate(d,30);
    a += rotate(b,5) + f3(c,d,e) + words[offset+4] + 0x8f1bbcdc; c = rotate(c,30);
  }

  for (int i = 12; i < 16; i++)
  {
    int offset = 5*i;
    e += rotate(a,5) + f2(b,c,d) + words[offset  ] + 0xca62c1d6; b = rotate(b,30);
    d += rotate(e,5) + f2(a,b,c) + words[offset+1] + 0xca62c1d6; a = rotate(a,30);
    c += rotate(d,5) + f2(e,a,b) + words[offset+2] + 0xca62c1d6; e = rotate(e,30);
    b += rotate(c,5) + f2(d,e,a) + words[offset+3] + 0xca62c1d6; d = rotate(d,30);
    a += rotate(b,5) + f2(c,d,e) + words[offset+4] + 0xca62c1d6; c = rotate(c,30);
  }

  m_hash[0] += a;
  m_hash[1] += b;
  m_hash[2] += c;
  m_hash[3] += d;
  m_hash[4] += e;
}

void SHA1::add(const void* data, size_t numBytes)
{
  const uint8_t* current = (const uint8_t*) data;

  if (m_bufferSize > 0)
  {
    while (numBytes > 0 && m_bufferSize < BlockSize)
    {
      m_buffer[m_bufferSize++] = *current++;
      numBytes--;
    }
  }

  if (m_bufferSize == BlockSize)
  {
    processBlock((void*)m_buffer);
    m_numBytes  += BlockSize;
    m_bufferSize = 0;
  }
```

```
  if (numBytes == 0)
    return;

  while (numBytes >= BlockSize)
  {
    processBlock(current);
    current    += BlockSize;
    m_numBytes += BlockSize;
    numBytes   -= BlockSize;
  }

  while (numBytes > 0)
  {
    m_buffer[m_bufferSize++] = *current++;
    numBytes--;
  }
}


void SHA1::processBuffer()
{
  size_t paddedLength = m_bufferSize * 8;

  paddedLength++;

  size_t lower11Bits = paddedLength & 511;
  if (lower11Bits <= 448)
    paddedLength +=       448 - lower11Bits;
  else
    paddedLength += 512 + 448 - lower11Bits;

  paddedLength /= 8;

  unsigned char extra[BlockSize];

  if (m_bufferSize < BlockSize)
    m_buffer[m_bufferSize] = 128;
  else
    extra[0] = 128;

  size_t i;
  for (i = m_bufferSize + 1; i < BlockSize; i++)
    m_buffer[i] = 0;
  for (; i < paddedLength; i++)
    extra[i - BlockSize] = 0;

  uint64_t msgBits = 8 * (m_numBytes + m_bufferSize);
```

```
  unsigned char* addLength;
  if (paddedLength < BlockSize)
    addLength = m_buffer + paddedLength;
  else
    addLength = extra + paddedLength - BlockSize;

  *addLength++ = (unsigned char)((msgBits >> 56) & 0xFF);
  *addLength++ = (unsigned char)((msgBits >> 48) & 0xFF);
  *addLength++ = (unsigned char)((msgBits >> 40) & 0xFF);
  *addLength++ = (unsigned char)((msgBits >> 32) & 0xFF);
  *addLength++ = (unsigned char)((msgBits >> 24) & 0xFF);
  *addLength++ = (unsigned char)((msgBits >> 16) & 0xFF);
  *addLength++ = (unsigned char)((msgBits >>  8) & 0xFF);
  *addLength   = (unsigned char)( msgBits       & 0xFF);

  processBlock(m_buffer);

  if (paddedLength > BlockSize)
    processBlock(extra);
}

std::string SHA1::getHash()
{
  unsigned char rawHash[HashBytes];
  getHash(rawHash);

  std::string result;
  result.reserve(2 * HashBytes);
  for (int i = 0; i < HashBytes; i++)
  {
    static const char dec2hex[16+1] = "0123456789abcdef";
    result += dec2hex[(rawHash[i] >> 4) & 15];
    result += dec2hex[ rawHash[i]       & 15];
  }

  return result;
}


void SHA1::getHash(unsigned char buffer[SHA1::HashBytes])
{
  uint32_t oldHash[HashValues];
  for (int i = 0; i < HashValues; i++)
    oldHash[i] = m_hash[i];

  processBuffer();
```

```
  unsigned char* current = buffer;
  for (int i = 0; i < HashValues; i++)
  {
    *current++ = (m_hash[i] >> 24) & 0xFF;
    *current++ = (m_hash[i] >> 16) & 0xFF;
    *current++ = (m_hash[i] >>  8) & 0xFF;
    *current++ =  m_hash[i]        & 0xFF;

    m_hash[i] = oldHash[i];
  }
}

std::string SHA1::operator()(const void* data, size_t numBytes)
{
  reset();
  add(data, numBytes);
  return getHash();
}

std::string SHA1::operator()(const std::string& text)
{
  reset();
  add(text.c_str(), text.size());
  return getHash();
}
```

Erlang (Key Pair Authentication)

Last updated：2023-12-22 10:09:25

Overview

This document describes how to authenticate and manage your APIs through key pair authentication in Erlang.

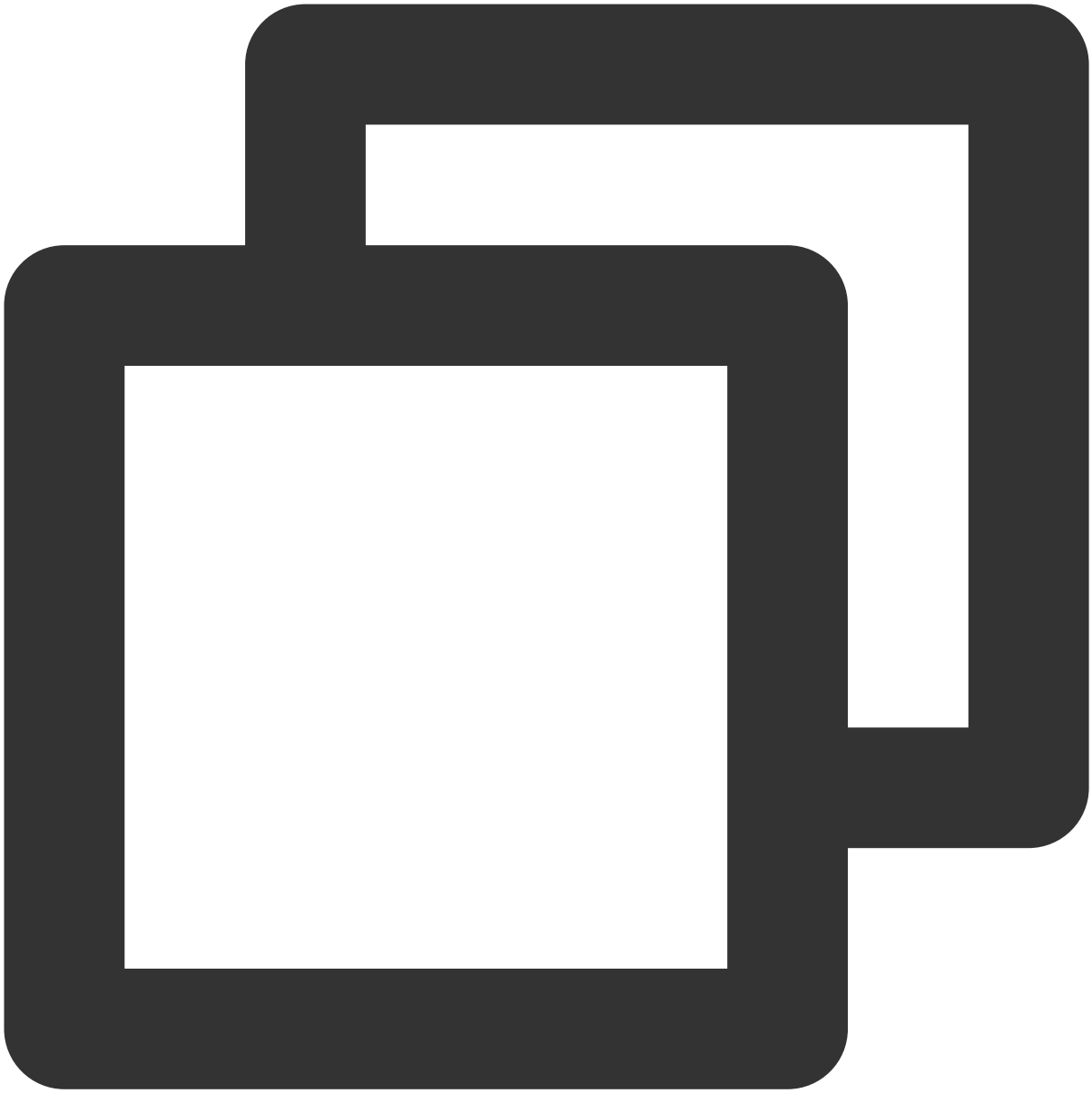
Directions

1. In the API Gateway console, create an API and select the authentication type as "key pair authentication" (for more information, please see API Creation Overview).
2. Publish the service where the API resides to the release environment (for more information, please see Service Release and Deactivation).
3. Create a key pair on the key management page in the console.
4. Create a usage plan on the usage plan page in the console and bind it to the created key pair (for more information, please see Sample Usage Plan).
5. Bind the usage plan to the API or the service where the API resides.
6. Generate signing information in Erlang by referring to the Sample Code.

Notes

The eventually delivered HTTP request contains at least two headers: `Date` or `X-Date` and `Authorization`. More optional headers can be added in the request. If `Date` is used, the server will not check the time; if `X-Date` is used, the server will check the time.
The value of the `Date` header is the construction time of the HTTP request in GMT format, such as Fri, 09 Oct 2015 00:00:00 GMT.
The value of the `X-Date` header is the construction time of the HTTP request in GMT format, such as Mon, 19 Mar 2018 12:08:40 GMT. The construction time of the HTTP request cannot deviate from the current time by more than 15 minutes.
If it is a microservice API, you need to add two fields in the header: `X-NameSpace-Code` and `X-MicroService-Name`. They are not needed for general APIs and are excluded in the demo.

Sample Code



```
-module(apigateway_erlang_demo).

%% API
-export([request_api/0]).

%% request_api()
request_api() ->

%%      Start the inets service. If the project has already started this service, r
        inets:start(),
```

```
        Url = "http://service-xxxxxxxx-1234567890.ap-beijing.apigateway.myqcloud.co
        Source = "xxxxxx",
        GMTDate = now_to_utc_string(),
        SecretId = "xxxDf4estwodtdzoke1234567890i3j9jv18wt9u",
        SecretKey = "xxxSNF0CEp3OhN4t91234567890AWrct960X9192",
        Sign = simple_sign(Source, GMTDate, SecretId, SecretKey),
        Header = [
                {"Source", Source},
                {"x-Date", GMTDate},
                {"Authorization", Sign},
                {"Content-Type", "application/x-www-form-urlencoded"}
        ],

        case httpc:request(get, {Url, Header}, [], []) of
                {ok, {_StatusLine, _Header, Result}} ->
                        Result;

%% %%                Result -> need decode

                Error ->
                        Error
        end.

simple_sign(Source, GMTDate, SecretId, SecretKey) ->
        Auth = "hmac id=\\"" ++ SecretId ++ "\\", algorithm=\\"hmac-sha1\\", header
        SecretKey = "xxxSNF0CEp3OhN4t91234567890AWrct960X9192",
        Source = "xxxxxx",
        SignStr = "x-date: " ++ GMTDate ++ "\\n" ++ "source: " ++ Source,
        Mac = crypto:hmac(sha, SecretKey, SignStr),
        Sign = base64:encode(Mac),
        Sign2 = binary_to_list(Sign),
        Sign3 = Auth ++ Sign2 ++ "\\"",
        Sign3.

%% Get the current time and convert it into the format of "Mon, 02 Jan 2006 15:04:0
now_to_utc_string() ->
        {{Year, Month, Day}, {Hour, Minute, Second}} = calendar:universal_time(),
        WeekNum = week_num(),
        Month1 = month_to_english(Month),
        WeekNum1 = week_to_english(WeekNum),
        Day1 = lists:flatten(io_lib:format("~2..0w", [Day])),
        Date1 = WeekNum1 ++ ", " ++ Day1 ++ " " ++ Month1,
        Date2 = lists:flatten(
                io_lib:format(" ~4..0w ~2..0w:~2..0w:~2..0w GMT",
                        [Year, Hour, Minute, Second])),
        Date1 ++ Date2.
```

```
week_num() ->
        {Date, _} = calendar:local_time(),
        calendar:day_of_the_week(Date).

%% day_of_the_week
week_to_english(1) ->
        "Mon";
week_to_english(2) ->
        "Tue";
week_to_english(3) ->
        "Wed";
week_to_english(4) ->
        "Thu";
week_to_english(5) ->
        "Fri";
week_to_english(6) ->
        "Sat";
week_to_english(7) ->
        "Sun".

month_to_english(1) ->
        "Jan";
month_to_english(2) ->
        "Feb";
month_to_english(3) ->
        "Mar";
month_to_english(4) ->
        "Apr";
month_to_english(5) ->
        "May";
month_to_english(6) ->
        "Jun";
month_to_english(7) ->
        "Jul";
month_to_english(8) ->
        "Aug";
month_to_english(9) ->
        "Sept";
month_to_english(10) ->
        "Oct";
month_to_english(11) ->
        "Nov";
month_to_english(12) ->
        "Dec".
```

Signature generation instructions

Last updated：2023-12-22 10:09:37

Precautions

If you select key pair authentication when creating an API, the service where the API resides needs to be published, the release environment needs to be bound to a usage plan, and the usage plan needs to be bound to a key pair. Then, the corresponding key pair will be used to generate signing information to authenticate API access requests.
For a `query` that contains Chinese characters and spaces, the request body should URL-encode the value with UTF-8.
When a signature is calculated through parameters, the value before encoding should be signed. URL-encoded strings cannot be signed. Please URL-encode the `query` and `body` values after signing.

Client Error Codes

Error Code	Status Code	Description	Solution
HMAC signature cannot be verified, a validate authorization header is required	401	The request did not carry the `Authorization` field.	Please construct this field as instructed in the signature demo for the applicable programming language.
authorization headers is invalidate	403	The format of the `Authorization` field in the request is incorrect.	Please construct this field as instructed in the signature demo for the applicable programming language.
id or signature missing	403	The ID or `signature` field is missing in the `Authorization` field of the request.	Please construct this field as instructed in the signature demo for the applicable programming language.
HMAC signature cannot be verified, a valid $header header is required	403	The `headers` field in the `Authorization` field of the request cannot be found in the request header.	Please construct this field as instructed in the signature demo for the applicable programming language.
HMAC signature cannot be verified, a valid date header is	403	The `Authorization` of the request must use the	Please construct this field as instructed in the signature

required		`date` field as one of the authentication fields.	demo for the applicable programming language.
Found no validate usage plan	403	Request authentication failed, because the API needs authentication, but the API is not bound to a usage plan.	Please disable API authentication or bind a usage plan to the release environment of the API and bind the key pair to the usage plan.
HMAC signature cannot be verified	403	Request authentication failed, because the `Key ID` in the `Authorization` field is not bound to the release environment of the API, the `Key ID` is invalid, or the `Key` is disabled.	Please check whether the key pair is available and bound to the corresponding usage plan/release environment.
HMAC signature does not match	403	Request authentication failed, because the calculated HMAC value does not match. Please check and calculate again.	The calculated HMAC value is incorrect. Please construct this field as instructed in the signature demo for the applicable programming language.
Not Found Host	404	The request does not have the `Host` field.	The request should have the `host` field, whose value should be the server's domain name in string type.
Get Host Fail	404	The value of the `host` field in the request is not in string type.	Please change the value of the `Host` field to string type.
Could not support method	404	The request method type is not supported.	Please check whether the request method is valid.
There is no api match host[$host]	404	Request server domain name/address not found.	Please check whether the request address is correct.
There is no api match env_mapping[$env_mapping]	404	The `env_mapping` field after the custom domain name is incorrect.	Please check whether the `env_mapping` configured for the bound custom domain name is the same as the one you entered.
There is no api match default env_mapping[$env_mapping]	404	The value of the `env_mapping` field after	The value of the `env_mapping` field after

		the default domain name should be `test/prepub/release` .	the default domain name should be `test/prepub/release` .
There is no api match uri[$uri]	404	The API that matches the URI is not found in the service corresponding to the request address.	Please check whether the entered URI is correct.
There is no api match method[$method]	404	The API corresponding to the request address plus URI does not support the request method.	Please check whether the request method is correct.
"Not allow use HTTPS protocol or Not allow use HTTP protocol"	404	The service corresponding to the requested address does not support the HTTP protocol type.	Please check whether the request protocol type is correct.
req is cross origin, api $uri need open cors flag on qcloud apigateway.	429	This is a cross-origin request, but the corresponding API has not enabled cross-origin access.	Please enable cross-origin access for the API.