ALIBABA CLOUD

Alibaba Cloud

API Gateway

Call API

Document Version: 20220614

Alibaba Cloud

Legal disclaimer

Document conventions

Style	Description	Example
⚠ Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	⚠ **Danger:** Resetting will result in the loss of user configuration data.
🔔 Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	🔔 **Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
🔈 Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	🔈 **Notice:** If the weight is set to 0, the server no longer receives new requests.
? Note	A note indicates supplemental instructions, best practices, tips, and other content.	? **Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings> Network> Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.Overview

This document describes how to call the APIs that you created and published in the API Gateway console by using code on your business system, such as a mobile app or an HTML page. This topic describes the following content: - Security authentication methods, use scenarios, and API calling methods by using code - API calling methods in different network environments - App authorization management by using an administrator account

1. Security authentication methods

- **No authentication**

APIs can be called on clients without authentication. This method is not secure and is suitable only for temporary tests. This method is not recommended.

- **AppCode for simple authentication**

This method is easy to use but is not secure. This method is suitable for scenarios where you make API calls between systems that are connected over an internal network. API Gateway checks whether the header or a query parameter in an API request from the client contains a valid AppCode for authentication. For more information, see Call an API operation by using an AppCode.

- **Digest authentication**

This method is more secure and suitable for more scenarios than the AppCode-based authentication method. If you use this authentication method to call a published API, the client uses the allocated AppKey and AppSecret to calculate the signature for the requested content. Then, the client transmits the key pair and signature in an HTTP request to API Gateway for verification.

- API Gateway SDKs are integrated with the signature mechanism. You can use the SDKs for different languages to call APIs. For more information, see Use SDKs to call APIs.

- If you want to calculate a signature on the client, see Request signature.

- **JWT-based authentication**

This method requires a JSON Web Token (JWT) authentication plug-in. This method provides the highest security and is suitable for scenarios that may cause security risks, such as JavaScript or web frontend development. For more information, see JWT authentication.

2. Network environments for API calls

Clients can initiate API requests over the Internet or a VPC. For more information about API calls over the Internet, see the "**Security authentication methods**" section.

For more information about API calls over a VPC, see the following topics:

- Access API Gateway over a VPC
- Access API Gateway from Function Compute over an internal network
- API calls implemented by API Gateway

3. Authorization management

If Security Certification is set to Alibaba Cloud APP when you create an API, you must authorize apps before they can call the API. You can create apps and then authorize them. You can also authorize the apps created by other users based on the app IDs. Apps are identities that you use to call APIs. Each app has a key pair that consists of an AppKey and an AppSecret. When you call an API in an app, the AppKey is specified as a header field in the request and the AppSecret is used to calculate a signature string that is attached to the request.

For more information, see Authorize an application to call an API operation.

2.Call an API

You can use API Gateway to call the API services enabled by other Alibaba Cloud users or third-party service providers. API Gateway provides for you a series of management services and support.

Call example

Based on the SDKs provided by API Gateway, you can write codes to call an API. You can also edit an HTTP request to call an API. The request structure of the API is as follows:

//If the domain name is a13db7999e494a90819cce500130034d.com.

//If the path is /web/cloudapi/mapping/service.

//If the query content is a=name, b=12.

//Then the URL of the request is as follows:

```
http://a13db7999e494a90819cce500130034d.com/web/cloudapi/mapping/service?a=name&amp;b=12
```

//Requesting method.

```
POST HttpMethod: POST
```

//Headers must include signature information and certain parameters.

//For more information about the methods of calculating and passing the encrypted signature, see Portal and Protocol.

```
X-Ca-Version:  1      // API version
X-Ca-Signature-Headers:  X-Ca-Version,X-Ca-Key,X-Ca-Stage,X-Ca-Timestamp      // Headers in
volved in signature calculation
X-Ca-Key:  60028305      //AppKey
X-Ca-Stage:  test      //Stage
X-Ca-Timestamp:  1456905123049       //Time stamp
X-Ca-Signature:  UAaH/qteir4G9UK4YR+NWdyq+c1rjl0PvtO/C1Qo68U=      // Signature
```

//Standard HTTP header.

```
Host:  a13db7999e494a90819cce500130034d.com         //Service address
Date:  Wed  02  Mar  2016  07:52:02  GMT
User-Agent:  Apache-HttpClient/4.1.2  (java  1.6)
Content-Type:  application/x-www-form-urlencoded;  charset=utf-8
```

//Body content.

```
Amount=11&InstanceId=ClientInstanceId&InstanceName=ClientInstanceName
```

An API request is constructed through the preceding content and the inputted parameters of the API. At the public beta stage, you must obtain API documentation and details, such as the service address and path, in the deprecation environment from the API service provider. The AppKey is the key for the created app, which is used for identity verification. The app is your identity to call an API. For more information, see subsequent content.

3.Authorize an application to call an API operation

Authorization means granting an application the permission to call an API operation. To call API operations, applications must be authorized.

- When you purchase an API operation in Alibaba Cloud Marketplace, you can authorize your own applications to call the API operation. If you do not have your own application, you can use the application that is automatically created and authorized for you to call the API operation. For more information, go to the Alibaba Cloud Marketplace console.
- If you and a partner reach an agreement that you can use an API operation that is provided by the partner, authorization must be performed by your partner, namely, the API provider. You must create your own application and provide the ID of the application for the API provider to authorize your application.

4.Call an API operation by using an AppCode

API Gateway provides various identity authentication methods for verifying API requests, including Alibaba Cloud APP and OpenID Connect. The Alibaba Cloud APP method supports two authentication modes: encrypted signature identity authentication and simple identity authentication.

1. Overview

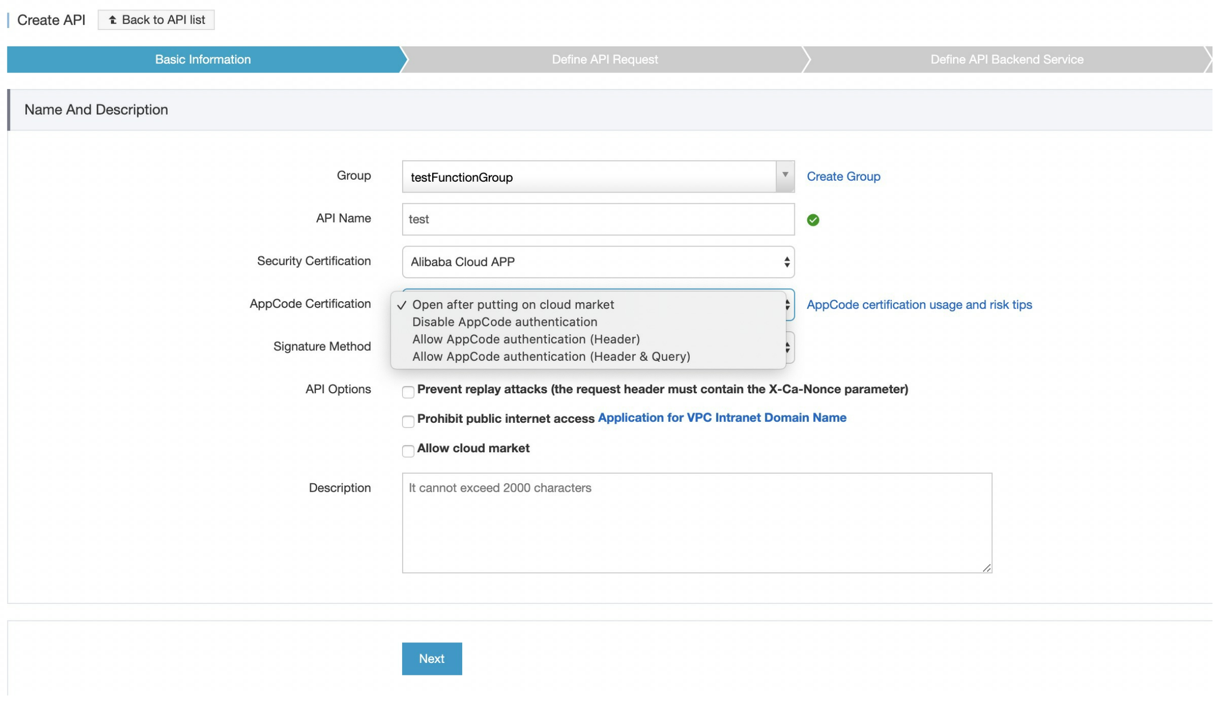
For information about encrypted signature identity authentication, see Request signature.

This topic describes in detail how to configure simple identity authentication and how to use this method to call an API operation. Simple identity authentication is much easier to implement than encrypted signature identity authentication. Simple identity authentication does not involve the complex process of signature calculation. Simple identity authentication requires only an AppCode that is automatically provided by API Gateway for an application. You must add the AppCode in the header or as a query parameter of an API request. The following process describes how to call an API operation by using simple identity authentication:

1. An API provider creates an API operation. When the API provider configures basic information for the API operation, the API provider sets the Security Certification parameter to Alibaba Cloud APP and select Allow AppCode authentication. By default, all API operations in Alibaba Cloud Marketplace support simple identity authentication.
2. An API caller creates an application in the API Gateway console. If the API caller purchases an API operation in Alibaba Cloud Marketplace, the API caller can also use the application that is automatically created and authorized for the API caller to call the API operation.
3. The API provider authorizes the API caller's application to call the API operation. For more information about application authorization, see Create an API operation with HTTP as the backend service.
4. The API caller logs on to the API Gateway console and performs the following steps: In the left-side navigation pane, choose Consume APIs > APPs. On the APP List page, find the target application and click the application name. The APP details page displays the AppCode that is used for simple identity authentication. Use the AppCode to call the API operation.

2. Create an API operation that uses simple identity authentication

1. When the API provider creates an API operation, the Security Certification parameter must be set to Alibaba Cloud APP or OpenID Connect & Alibaba Cloud APP.
2. The API provider needs to set the AppCode Certification parameter to Allow AppCode authentication (Header) or Allow AppCode authentication (Header & Query).



Create API | Back to API list

Basic Information | Define API Request | Define API Backend Service

Name And Description

Group	testFunctionGroup
API Name	test
Security Certification	Alibaba Cloud APP
AppCode Certification	✓ Open after putting on cloud market Disable AppCode authentication Allow AppCode authentication (Header) Allow AppCode authentication (Header & Query)
Signature Method	
API Options	☐ Prevent replay attacks (the request header must contain the X-Ca-Nonce parameter) ☐ Prohibit public internet access Application for VPC Intranet Domain Name ☐ Allow cloud market
Description	It cannot exceed 2000 characters

As shown in the figure, the AppCode Certification parameter has the following valid values:

- Open after putting on cloud market: By default, AppCode authentication is disabled. After the API operation is available in Alibaba Cloud Marketplace, AppCode authentication is enabled. The AppCode of an application must be added in the header of each API request.
- Disable AppCode authentication: AppCode authentication is disabled no matter whether the API operation is available in Alibaba Cloud Marketplace. The API operation can be called only by using encrypted signature identity authentication.
- Allow AppCode authentication (Header): AppCode authentication is enabled no matter whether the API operation is available in Alibaba Cloud Marketplace. The AppCode of an application must be added in the header of each API request.
- Allow AppCode authentication (Header & Query): AppCode authentication is enabled no matter whether the API operation is available in Alibaba Cloud Marketplace. The AppCode of an application can be added in the header or as a query parameter of each API request.

Note: When the API provider defines request parameters for the API operation, the API provider does not need to define a header parameter or a query parameter for the AppCode.

3. Call an API operation by using simple identity authentication

Before an API provider provides an API operation for an API caller, the API provider must ensure that AppCode authentication is enabled for the API operation. Then, the API caller can use simple identity authentication to call the API operation without the need to implement complex signature algorithms on a client. This section describes how to call an API operation by using simple identity authentication. When you call an API operation in an application by using simple identity authentication, you can add the AppCode of your application in the header or as a query parameter of the API request.

3.1 Add the AppCode in the header

- Add a field named Authorization to the header of an API request.
- The value of the Authorization field must be in the following format: APPCODE + Space + AppCode value.

Format:

```
Authorization:APPCODE AppCode value
```

Example:

```
Authorization:APPCODE 3F2504E04F8911D39A0C0305E82C3301
```

3.2 Add the AppCode as a query parameter

- Add the AppCode as a query parameter of an API request. The name of the parameter can be appcode, appCode, APPCODE, or APPCode.
- The value of the parameter is the value of the AppCode.

Example:

```
http://www.aliyum.com?AppCode=3F2504E04F8911D39A0C0305E82C3301
```

4. Risk warning

Simple identity authentication is easy to implement because it does not involve the complex process of signature calculation. However, transmitting the AppCode as plaintext on networks may cause security risks.

You must make sure that your client and API Gateway communicate with each other by using HTTPS instead of HTTP or the WebSocket protocol. Neither HTTP nor the WebSocket protocol supports encryption. Therefore, an AppCode will be transmitted as plaintext if you use HTTP or the WebSocket protocol. This causes a high risk that the AppCode may be captured by hackers.

5.Use SDKs to call APIs

This topic describes how to use the SDKs that are automatically generated in API Gateway to call published APIs in your business system.

Overview

The SDKs that are automatically generated in API Gateway are integrated with a signature mechanism. By using the signature mechanism, you do not need to calculate a signature. If you want to calculate a signature on the client, see Request signature.

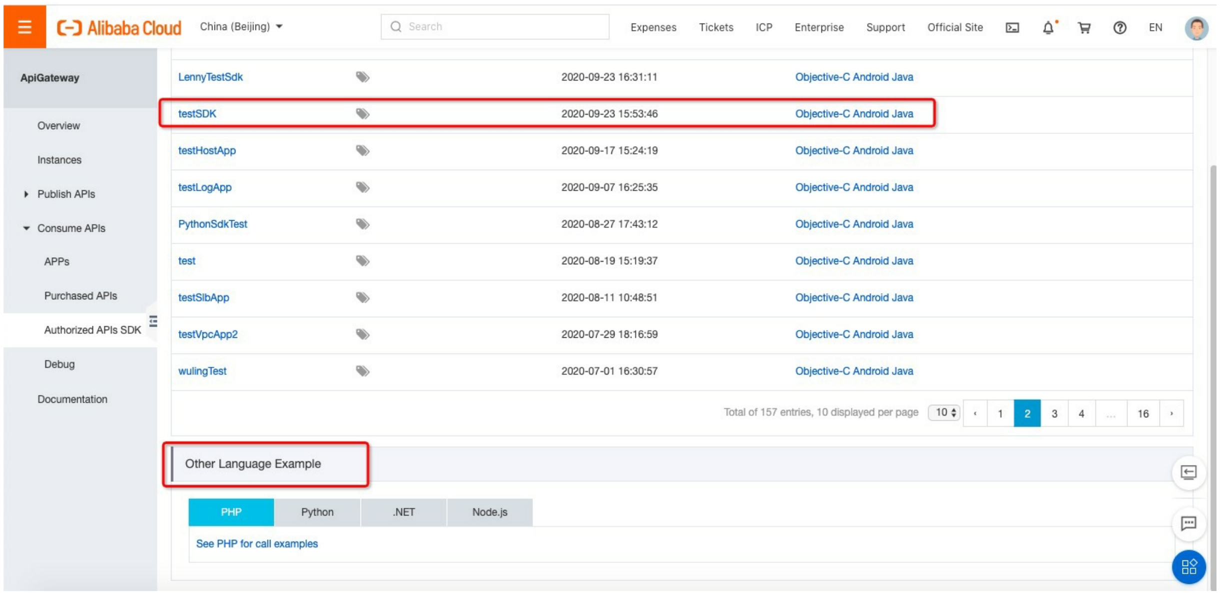
For information about the preparations made to call APIs, see Create an API operation with HTTP as the backend service. The APIs that you want to call by using SDKs must meet the following conditions:

- The APIs use one of the following security certification methods: Alibaba Cloud APP and OpenID Connect & Alibaba Cloud APP.
- Apps are authorized before they can call the APIs.
- The APIs are published to a Release environment.

1. Download SDKs

You can use one of the following methods to download SDKs in the API Gateway console:

1.1 In the left-side navigation pane of the console, choose `Consume APIs` > `Authorized APIs SDK`. On the Authorized APIs SDK Auto-Generation page, download the app-specific SDKs that are used to call APIs. API Gateway automatically generates SDKs for Objective-C, Android, and Java. For information about SDKs for other languages, see the Other Language Example section on the Authorized APIs SDK Auto-Generation page.



LennyTestSdk	2020-09-23 16:31:11	Objective-C Android Java
testSDK	2020-09-23 15:53:46	Objective-C Android Java
testHostApp	2020-09-17 15:24:19	Objective-C Android Java
testLogApp	2020-09-07 16:25:35	Objective-C Android Java
PythonSdkTest	2020-08-27 17:43:12	Objective-C Android Java
test	2020-08-19 15:19:37	Objective-C Android Java
testSlbApp	2020-08-11 10:48:51	Objective-C Android Java
testVpcApp2	2020-07-29 18:16:59	Objective-C Android Java
wulingTest	2020-07-01 16:30:57	Objective-C Android Java

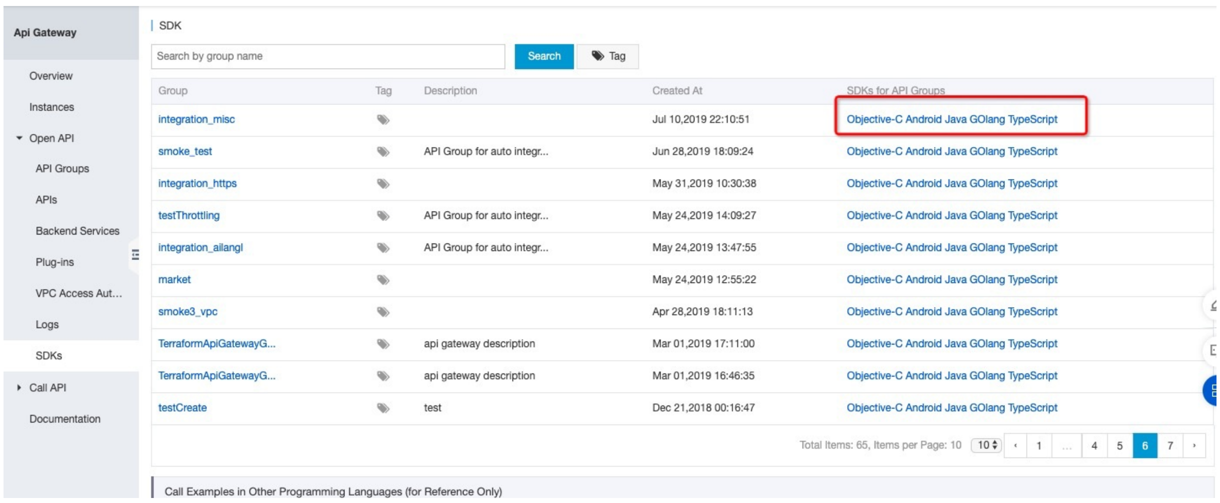
Total of 157 entries, 10 displayed per page 10 ‹ 1 2 3 4 ... 16 ›

Other Language Example

PHP Python .NET Node.js

See PHP for call examples

1.2 In the left-side navigation pane of the console, choose `Publish APIs` > `Owned APIs SDK`. On the Owned APIs SDK Auto-Generation page, download API group-specific SDKs. API Gateway automatically generates SDKs for Objective-C, Android, Java, Golang, and TypeScript.



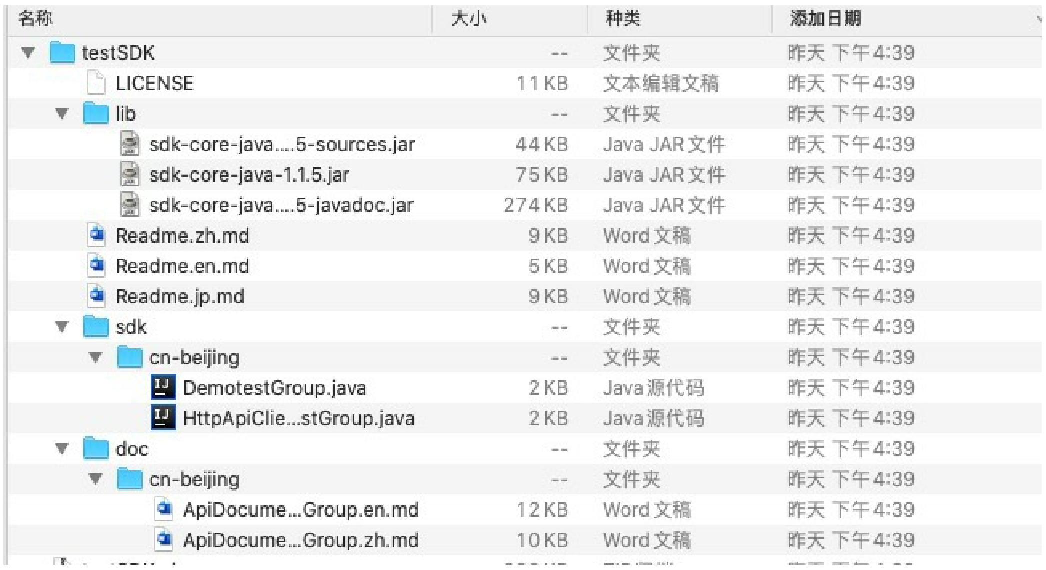
Group	Tag	Description	Created At	SDKs for API Groups
integration_misc			Jul 10,2019 22:10:51	Objective-C Android Java GOlang TypeScript
smoke_test		API Group for auto integr...	Jun 28,2019 18:09:24	Objective-C Android Java GOlang TypeScript
integration_https			May 31,2019 10:30:38	Objective-C Android Java GOlang TypeScript
testThrottling		API Group for auto integr...	May 24,2019 14:09:27	Objective-C Android Java GOlang TypeScript
integration_ailangl		API Group for auto integr...	May 24,2019 13:47:55	Objective-C Android Java GOlang TypeScript
market			May 24,2019 12:55:22	Objective-C Android Java GOlang TypeScript
smoke3_vpc			Apr 28,2019 18:11:13	Objective-C Android Java GOlang TypeScript
TerraformApiGatewayG...		api gateway description	Mar 01,2019 17:11:00	Objective-C Android Java GOlang TypeScript
TerraformApiGatewayG...		api gateway description	Mar 01,2019 16:46:35	Objective-C Android Java GOlang TypeScript
testCreate		test	Dec 21,2018 00:16:47	Objective-C Android Java GOlang TypeScript

Total Items: 65, Items per Page: 10 10 ‹ 1 ... 4 5 6 7 ›

Call Examples in Other Programming Languages (for Reference Only)

2. Use SDKs to call APIs

The following figure shows the directory structure of a decompressed SDK package that you downloaded in the API Gateway console.



名称	大小	种类	添加日期
testSDK	--	文件夹	昨天 下午4:39
LICENSE	11 KB	文本编辑文稿	昨天 下午4:39
lib	--	文件夹	昨天 下午4:39
sdk-core-java....5-sources.jar	44 KB	Java JAR文件	昨天 下午4:39
sdk-core-java-1.1.5.jar	75 KB	Java JAR文件	昨天 下午4:39
sdk-core-java....5-javadoc.jar	274 KB	Java JAR文件	昨天 下午4:39
Readme.zh.md	9 KB	Word文稿	昨天 下午4:39
Readme.en.md	5 KB	Word文稿	昨天 下午4:39
Readme.jp.md	9 KB	Word文稿	昨天 下午4:39
sdk	--	文件夹	昨天 下午4:39
cn-beijing	--	文件夹	昨天 下午4:39
DemotestGroup.java	2 KB	Java源代码	昨天 下午4:39
HttpApiClie...stGroup.java	2 KB	Java源代码	昨天 下午4:39
doc	--	文件夹	昨天 下午4:39
cn-beijing	--	文件夹	昨天 下午4:39
ApiDocume...Group.en.md	12 KB	Word文稿	昨天 下午4:39
ApiDocume...Group.zh.md	10 KB	Word文稿	昨天 下午4:39

- sdk folder

* sdk/{{regionId}} `SDK for Java folder, which contains the code to call all APIs in each group.`

* HttpApiClient{{group}}.java `Describes how to call APIs in a specific group over HTTP.`

* HttpsApiClient{{group}}.java `Describes how to call APIs in a specific group over HTTPS.`

* WebSocketApiClient{{group}}.java `Describes how to call APIs in a specific group over WebSocket.`

* Demo{{group}}.java `Contains the sample code of all API calls in a specific group.`

- doc/{{regionId}} folder

* ApiDocument_{{group}}.md `Indicates the API documentation in a specific group.`

- lib folder

* sdk-core-java-1.1.5.jar `Core package, which is the dependency of the SDK.`

* sdk-core-java-1.1.5-sources.jar `Source code of the dependency.`

* sdk-core-java-1.1.5-javadoc.jar `JAR file of the core package.`

Readme.md `User guide of the SDK.`

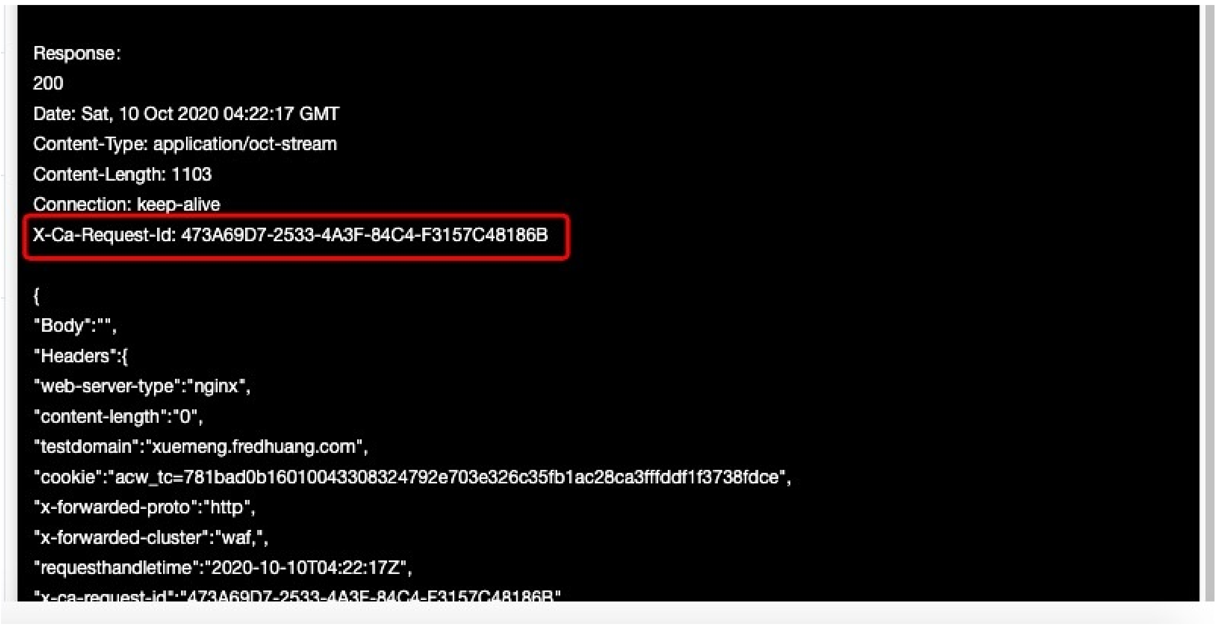
LICENSE `License`

For more information about how to use SDKs, see **Readme.md**.

Note: AppKey and AppSecret are used to authenticate user requests in API Gateway. If AppKey and AppSecret are stored on the client, encrypt them.

3. Troubleshooting

If an error occurs when you use an SDK to call an API, you can obtain the request ID from the X-Ca-Request-Id header and troubleshoot the issue. For more information about troubleshooting, see Troubleshooting.



```
Response:
200
Date: Sat, 10 Oct 2020 04:22:17 GMT
Content-Type: application/oct-stream
Content-Length: 1103
Connection: keep-alive
X-Ca-Request-Id: 473A69D7-2533-4A3F-84C4-F3157C48186B

{
"Body":"",
"Headers":{
"web-server-type":"nginx",
"content-length":"0",
"testdomain":"xuemeng.fredhuang.com",
"cookie":"acw_tc=781bad0b16010043308324792e703e326c35fb1ac28ca3fffddf1f3738fdce",
"x-forwarded-proto":"http",
"x-forwarded-cluster":"waf,",
"requesthandletime":"2020-10-10T04:22:17Z",
"x-ca-request-id":"473A69D7-2533-4A3F-84C4-F3157C48186B"
```

6.Request signature

To call a published API that uses the digest authentication method (AppKey and AppSecret), the client must use the signature key pair to calculate the signature for the requested content, and then transmit the signature to the server for verification. The SDKs for API Gateway are integrated with the signature mechanism. After the signature key pair is configured in the SDKs, you can initiate a request that carries the correct signature. This topic describes how to calculate signatures on the client.

1. Principle of signature and signature verification

API Gateway provides the signature and signature verification functions, which are used to:

- Check the validity of requests received from clients to ensure that the requests contain the correct signature that is generated based on the authorized AppKey.
- Prevent the requested data from being tampered with during transmission.

The API provider can create an app on the APPs page of the API Gateway console. Each app carries a signature key pair consisting of AppKey and AppSecret. After the API provider authorizes the app to use the API, the API caller can use the signature key pair to call the API. Note that the app can be issued by the API provider or owned by the API caller.

When the client calls the API, it must use the authorized signature key pair to encrypt and sign key data in the request, and includes the AppKey and the signature string in the header of the request transmitted to API Gateway. After receiving the request, API Gateway reads the AppKey from the request header, obtains the AppSecret that corresponds to the AppKey, uses the AppSecret to calculate the signature of key data in the received request, and compares the calculated signature with that received from the client to verify the signature. Only requests that pass signature verification are sent to backend services. If API Gateway determines that a request is invalid, it returns an error.

API Gateway verifies only signatures contained in API requests that use the following security certification methods: Alibaba Cloud APP and OpenID Connect & Alibaba Cloud APP.

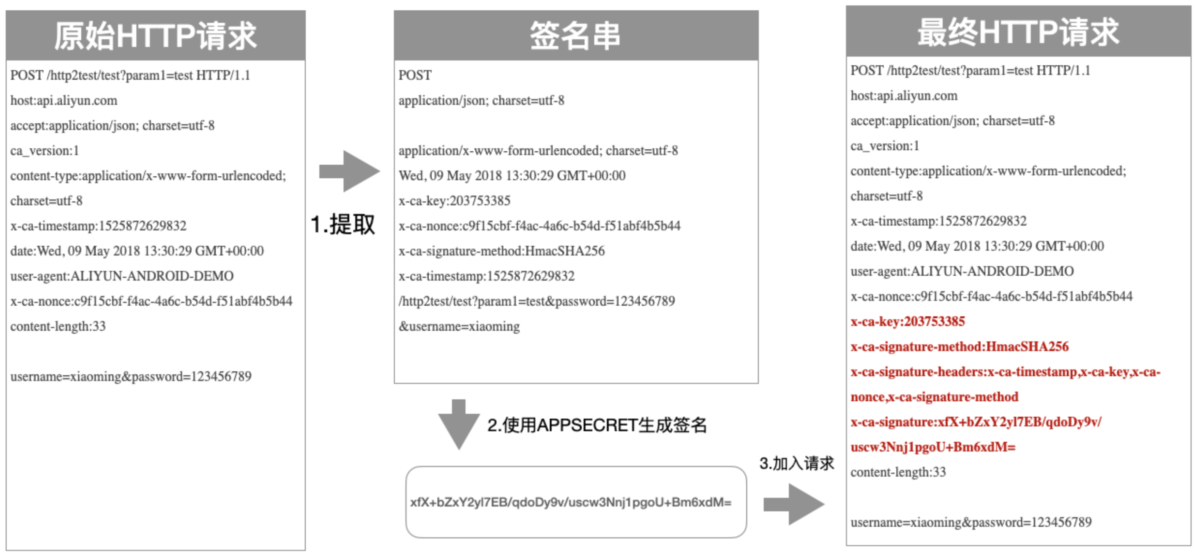
1.1 Prerequisites

The API caller has obtained the signature key pair assigned to the related API before calling the API.

- APP Key
- APP Secret

The called API uses the following security certification methods: Alibaba Cloud APP and OpenID Connect & Alibaba Cloud APP.

1.2 Signature generation on the client



```
POST /http2test/test?param1=test HTTP/1.1
host:api.aliyun.com
accept:application/json; charset=utf-8
ca_version:1
content-type:application/x-www-form-urlencoded;
charset=utf-8
x-ca-timestamp:1525872629832
date:Wed, 09 May 2018 13:30:29 GMT+00:00
user-agent:ALIYUN-ANDROID-DEMO
x-ca-nonce:c9f15cbf-f4ac-4a6c-b54d-f51abf4b5b44
content-length:33

username=xiaoming&password=123456789
```

```
POST
application/json; charset=utf-8

application/x-www-form-urlencoded; charset=utf-8
Wed, 09 May 2018 13:30:29 GMT+00:00
x-ca-key:203753385
x-ca-nonce:c9f15cbf-f4ac-4a6c-b54d-f51abf4b5b44
x-ca-signature-method:HmacSHA256
x-ca-timestamp:1525872629832
/http2test/test?param1=test&password=123456789
&username=xiaoming
```

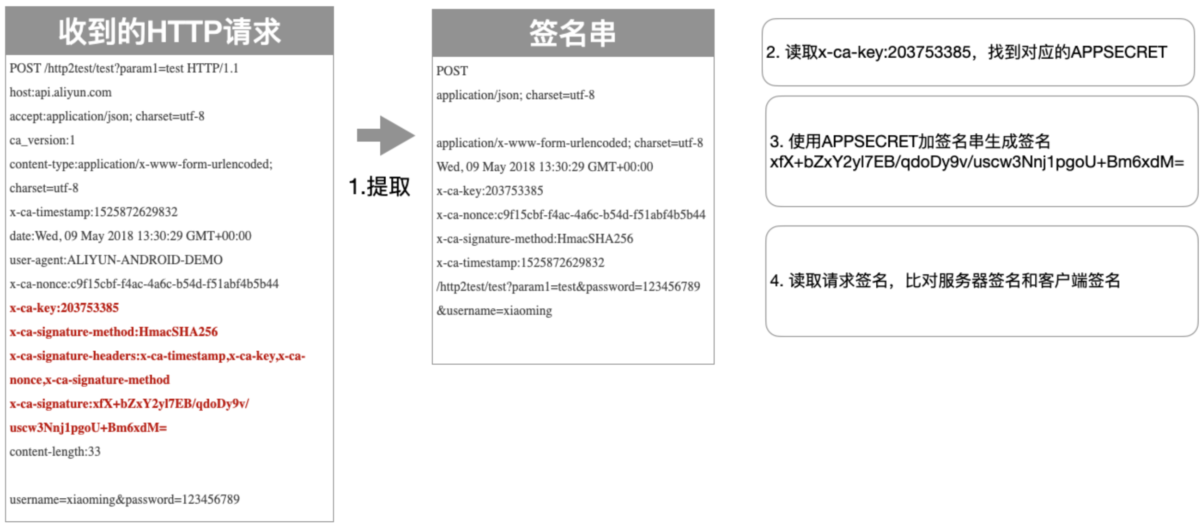
```
xfX+bZxY2yl7EB/qdoDy9v/uscw3Nnj1pgoU+Bm6xdM=
```

```
POST /http2test/test?param1=test HTTP/1.1
host:api.aliyun.com
accept:application/json; charset=utf-8
ca_version:1
content-type:application/x-www-form-urlencoded;
charset=utf-8
x-ca-timestamp:1525872629832
date:Wed, 09 May 2018 13:30:29 GMT+00:00
user-agent:ALIYUN-ANDROID-DEMO
x-ca-nonce:c9f15cbf-f4ac-4a6c-b54d-f51abf4b5b44
x-ca-key:203753385
x-ca-signature-method:HmacSHA256
x-ca-signature-headers:x-ca-timestamp,x-ca-key,x-ca-
nonce,x-ca-signature-method
x-ca-signature:xfX+bZxY2yl7EB/qdoDy9v/
uscw3Nnj1pgoU+Bm6xdM=
content-length:33

username=xiaoming&password=123456789
```

The client generates a signature in three steps:

1. Extract key data from the original request to obtain a signature string.
2. Use the encryption algorithm and AppSecret to encrypt the signature string to obtain the signature.
3. Add headers related to the signature to the original HTTP request to obtain the final HTTP request.

1.3 Signature verification on the server



```
POST /http2test/test?param1=test HTTP/1.1
host:api.aliyun.com
accept:application/json; charset=utf-8
ca_version:1
content-type:application/x-www-form-urlencoded;
charset=utf-8
x-ca-timestamp:1525872629832
date:Wed, 09 May 2018 13:30:29 GMT+00:00
user-agent:ALIYUN-ANDROID-DEMO
x-ca-nonce:c9f15cbf-f4ac-4a6c-b54d-f51abf4b5b44
x-ca-key:203753385
x-ca-signature-method:HmacSHA256
x-ca-signature-headers:x-ca-timestamp,x-ca-key,x-ca-
nonce,x-ca-signature-method
x-ca-signature:xfX+bZxY2yl7EB/qdoDy9v/
uscw3Nnj1pgoU+Bm6xdM=
content-length:33

username=xiaoming&password=123456789
```

```
POST
application/json; charset=utf-8

application/x-www-form-urlencoded; charset=utf-8
Wed, 09 May 2018 13:30:29 GMT+00:00
x-ca-key:203753385
x-ca-nonce:c9f15cbf-f4ac-4a6c-b54d-f51abf4b5b44
x-ca-signature-method:HmacSHA256
x-ca-timestamp:1525872629832
/http2test/test?param1=test&password=123456789
&username=xiaoming
```

2. 读取x-ca-key:203753385，找到对应的APPSECRET

3. 使用APPSECRET加签名串生成签名
xfX+bZxY2yl7EB/qdoDy9v/uscw3Nnj1pgoU+Bm6xdM=

4. 读取请求签名，比对服务器签名和客户端签名

The server verifies the signature received from the client in four steps:

1. Extract key data from the request to obtain the signature string.
2. Read the AppKey from the request and use the AppKey to obtain the AppSecret.
3. Use the encryption algorithm and AppSecret to encrypt the signature string to obtain the signature.
4. Read the client-side signature from the request and compare the server-side signature with the client-side signature.

2. Signature generation and transfer

2.1 Extract the signature string

The client must extract key data from the HTTP requests and combine the data into a signature string. The signature string is in the following format:

```
HTTPMethod
Accept
Content-MD5
Content-Type
Date
Headers
PathAndParameters
```

The preceding seven fields constitute a signature string and are separated with \n. If the Headers field is left empty, \n is not required. If the other fields are left empty, \n must be retained. The signature is case-sensitive. Data extraction for each field follows these rules:

- HTTPMethod: the HTTP method used to send a request, such as POST. The field value is in uppercase.
- Accept: the value of the Accept header in the request, which can be left empty. We recommend that the Accept header be explicitly displayed in the request. If the Accept header is empty, some HTTP clients will use the default value */* for Accept, causing signature verification to fail.
- Content-MD5: the value of the Content-MD5 header in a request. This field can be left empty. The value is calculated only when a request contains a body that is not of the Form type. The following example shows how to calculate the value of the Content-MD5 header by using Java:

```
String content-MD5 = Base64.encodeBase64(MD5(bodyStream.getbytes("UTF-8")));
```

- Content-Type: the value of the Content-Type header, which can be left empty.
- Date: the value of the Date header in the request, which can be left empty.
- Headers: You can specify the headers for signature calculation. The following rules apply when you concatenate signature strings:
  - Header keys are first sorted in alphabetical order and then concatenated in the following way:

    ```
    HeaderKey1 + ":" + HeaderValue1 + "\n"\+
    HeaderKey2 + ":" + HeaderValue2 + "\n"\+
    ...
    HeaderKeyN + ":" + HeaderValueN + "\n"
    ```

  - If the value of a header is empty, use HeaderKey + ":" + "\n" for signature and retain the header key and colon.
  - The header keys used for signature calculation are separated with commas (,) and placed in headers whose key is X-Ca-Signature-Headers.
  - The following headers are not used for signature calculation: X-Ca-Signature, X-Ca-Signature-Headers, Accept, Content-MD5, Content-Type, and Date.
- The PathAndParameters field contains all the parameters of the path, query, and form types.

```
Path + "?" + Key1 + "=" + Value1 + "&" + Key2 + "=" + Value2 + ... "&" + KeyN + "=" + Val
ueN
```

- Keys of query and form parameters are first sorted in alphabetical order and then concatenated by using the preceding method.
- If the query and form parameters are left empty, you can use path parameters without the need to add the question mark (?).
- If the parameter value is empty, only the key is used for signature calculation.
- If query and form parameters have array parameters (parameters with the same key but different values), the first value is used for signature calculation.

The following example shows parameters in a common HTTP request:

```
POST /http2test/test? param1=test HTTP/1.1
host:api.aliyun.com
accept:application/json; charset=utf-8
ca_version:1
content-type:application/x-www-form-urlencoded; charset=utf-8
x-ca-timestamp:1525872629832
date:Wed, 09 May 2018 13:30:29 GMT+00:00
user-agent:ALIYUN-ANDROID-DEMO
x-ca-nonce:c9f15cbf-f4ac-4a6c-b54d-f51abf4b5b44
content-length:33
username=xiaoming&password=123456789
```

The following example shows a correct signature string:

```
POST
application/json; charset=utf-8
application/x-www-form-urlencoded; charset=utf-8
Wed, 09 May 2018 13:30:29 GMT+00:00
x-ca-key:203753385
x-ca-nonce:c9f15cbf-f4ac-4a6c-b54d-f51abf4b5b44
x-ca-signature-method:HmacSHA256
x-ca-timestamp:1525872629832
/http2test/test? param1=test&password=123456789&username=xiaoming
```

2.2 Signature calculation

After the client combines key data extracted from the HTTP request into a signature string, it must encrypt and encode the signature string to generate the final signature. API Gateway supports the following encryption algorithms:

- HmacSHA256
- HmacSHA1

The following example shows the encryption method. stringToSign is the extracted signature string and secret is the AppSecret.

```
Mac hmacSha256 = Mac.getInstance("HmacSHA256");
hmacSha256.init(new SecretKeySpec(secret.getBytes("UTF-8"), 0, keyBytes.length, "HmacSHA256
"));
byte[] md5Result = hmacSha1.doFinal(stringToSign.getBytes("UTF-8"));
String sign = Base64.encodeBase64String(md5Result);
```

```
Mac hmacSha1 = Mac.getInstance("HmacSHA1");
hmacSha1.init(new SecretKeySpec(secret.getBytes("UTF-8"), 0, keyBytes.length, "HmacSHA1"));
byte[] md5Result = hmacSha1.doFinal(stringToSign.getBytes("UTF-8"));
String sign = Base64.encodeBase64String(md5Result);
```

First, a Byte array is obtained after StringToSign is UTF-8-decoded. The Byte array is then encrypted and Base64-encoded to generate the final signature.

2.3 Signature transfer

The client must include the following four headers in the requests sent to API Gateway for signature verification:

- x-ca-key: the AppKey, which is required.
- x-ca-signature-method: the signature method, which is optional. Valid values: HmacSHA256 and HmacSHA1. Default value: HmacSHA256.
- X-Ca-Signature-Headers: The keys of all signature headers. Separate multiple keys with commas (,). This parameter is optional.
- X-Ca-Signature: the signature, which is required.

The following example shows an HTTP request with a signature:

```
POST /http2test/test? param1=test HTTP/1.1
host:api.aliyun.com
accept:application/json; charset=utf-8
ca_version:1
content-type:application/x-www-form-urlencoded; charset=utf-8
x-ca-timestamp:1525872629832
date:Wed, 09 May 2018 13:30:29 GMT+00:00
user-agent:ALIYUN-ANDROID-DEMO
x-ca-nonce:c9f15cbf-f4ac-4a6c-b54d-f51abf4b5b44
x-ca-key:203753385
x-ca-signature-method:HmacSHA256
x-ca-signature-headers:x-ca-timestamp,x-ca-key,x-ca-nonce,x-ca-signature-method
x-ca-signature:xfX+bZxY2yl7EB/qdoDy9v/uscw3Nnj1pgoU+Bm6xdM=
content-length:33
username=xiaoming&password=123456789
```

3. Signature troubleshooting

If signature verification fails, API Gateway contains the signature string (StringToSign) of the server in the header of the HTTP response that is returned to the client. The key is X-Ca-Error-Message. You can find out errors by comparing the signature string (StringToSign) calculated on the client side with the signature string returned by the server.

If the two values are the same, check the AppSecret that is used for signature calculation.

HTTP headers do not support line breaks. Line breaks in stringToSign are replaced with number signs (#).

```
errorMessage:  Invalid Signature, Server StringToSign:`GET#application/json##application/js
on##X-Ca-Key:200000#X-Ca-Timestamp:1589458000000#/app/v1/config/keys? keys=TEST`
```

The following example shows the signature on the server side:

```
GET
application/json
application/json
X-Ca-Key:200000
X-Ca-Timestamp:1589458000000
/app/v1/config/keys? keys=TEST
```

4. Example of using Java for signature

For more information about signature calculation, see SDKs provided by API Gateway. You can download Java, Android, and Objective-C SDKs with source code on the following pages of the API Gateway console:

- Publish APIs - Owned APIs SDK Auto-Generation
- Consume APIs - Authorized APIs SDK Auto-Generation

For signature calculation by using Java and Android, see the class com.alibaba.cloudapi.sdk.util.SignUtil.

7.Access API Gateway over a VPC

You can access API Gateway either over the Internet or over an internal network. This topic describes how to access API Gateway over a VPC.

Instructions for using VPC subdomains

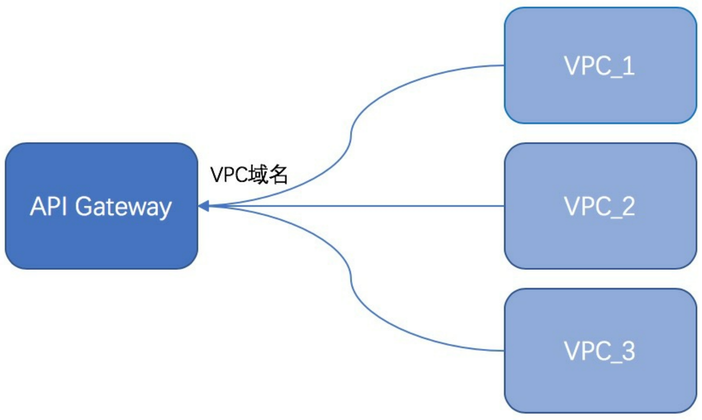
You can configure a VPC subdomain for each API group.

- This subdomain can be used to access APIs in an API group over a VPC. There is no such limit of 1,000 API calls per day.
- You cannot use a VPC subdomain to access APIs over HTTPS. To achieve HTTPS-based access, bind your domain name to the API group.
- To bind your domain name to the API group, add a CNAME record to resolve your domain name to the VPC subdomain.

The activation method and effective scope of a VPC subdomain vary according to instance types. For more information, see the following descriptions.

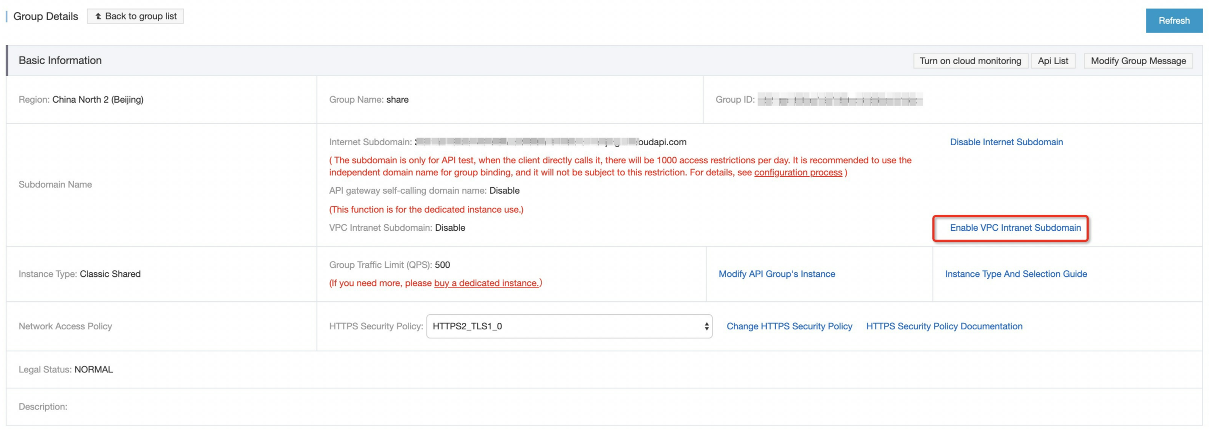
Internal endpoint of a shared instance

Users located in the same region can access a shared API Gateway instance over their own VPCs.



Procedure

1. Log on to the API Gateway console.
2. Navigate through Publish APIs > API Groups.
3. On the Group List page, find the target API group and click the group name.
4. On the Group Details page that appears, click Enable VPC Intranet Subdomain. API Gateway automatically assigns a VPC subdomain to this API group. You can use this domain to access required APIs in this API group.



Group Details ↥ Back to group list

Refresh

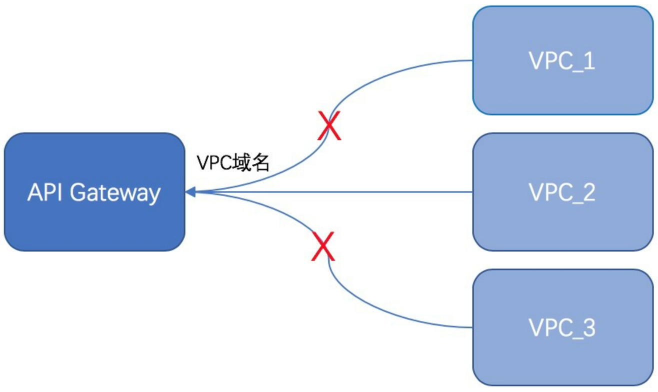
Basic Information

Turn on cloud monitoring | Api List | Modify Group Message

Region: China North 2 (Beijing)	Group Name: share	Group ID: [redacted]	
Subdomain Name	Internet Subdomain: [redacted]oudapi.com (The subdomain is only for API test, when the client directly calls it, there will be 1000 access restrictions per day. It is recommended to use the independent domain name for group binding, and it will not be subject to this restriction. For details, see configuration process) API gateway self-calling domain name: Disable (This function is for the dedicated instance use.) VPC Intranet Subdomain: Disable		Disable Internet Subdomain Enable VPC Intranet Subdomain
Instance Type: Classic Shared	Group Traffic Limit (QPS): 500 (If you need more, please buy a dedicated instance.)	Modify API Group's Instance	Instance Type And Selection Guide
Network Access Policy	HTTPS Security Policy: HTTPS2_TLS1_0	Change HTTPS Security Policy	HTTPS Security Policy Documentation
Legal Status: NORMAL			
Description:			

Internal endpoint of a dedicated instance

A dedicated API Gateway instance supports only one VPC. Users in other VPCs cannot access APIs configured for the instance. This access method is more secure.

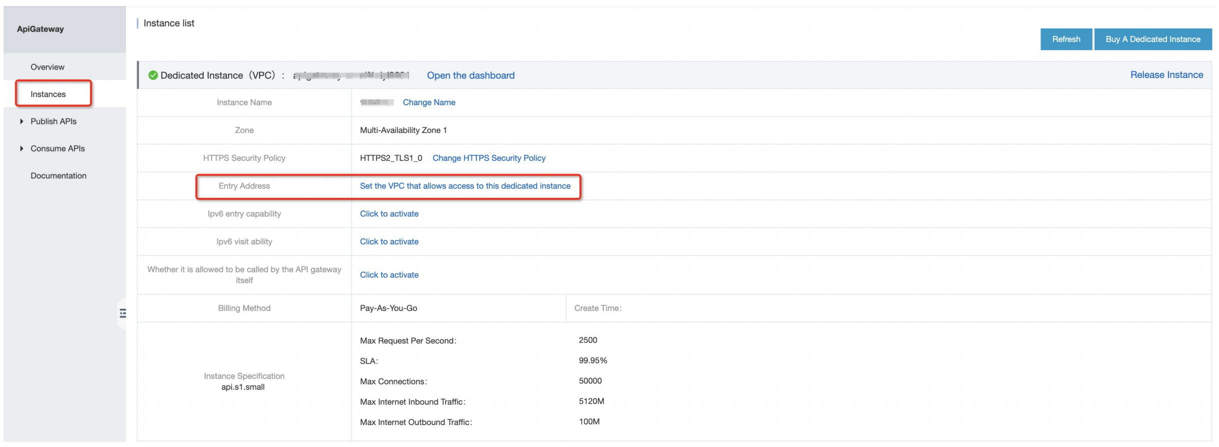


Procedure

1. Log on to the API Gateway console.

2. In the left-side navigation pane, click Instances.

3. Find the target instance and configure a VPC through which users can access APIs under the instance.



Instance list

Refresh | Buy A Dedicated Instance

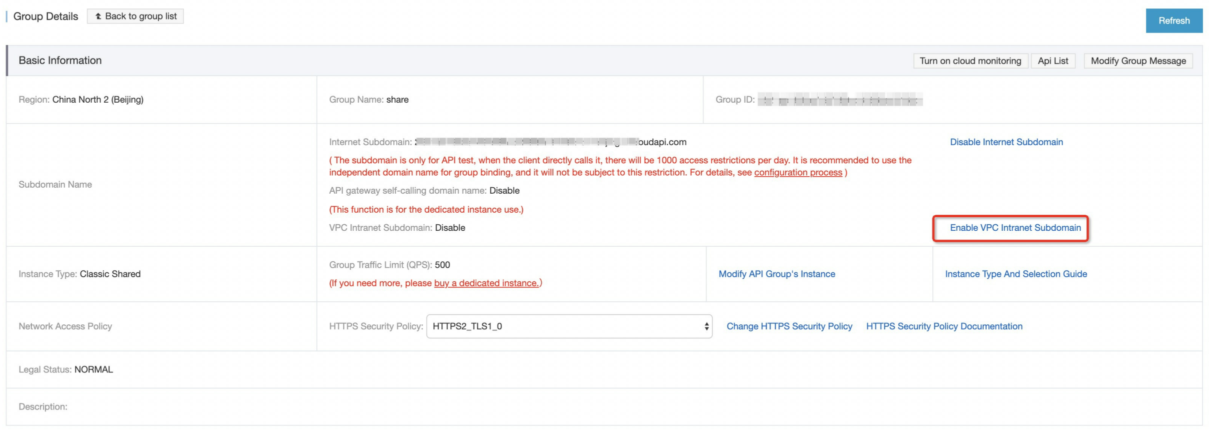
Dedicated Instance（VPC）： [redacted] Open the dashboard — Release Instance

Instance Name	[redacted] Change Name	
Zone	Multi-Availability Zone 1	
HTTPS Security Policy	HTTPS2_TLS1_0 Change HTTPS Security Policy	
Entry Address	Set the VPC that allows access to this dedicated instance	
Ipv6 entry capability	Click to activate	
Ipv6 visit ability	Click to activate	
Whether it is allowed to be called by the API gateway itself	Click to activate	
Billing Method	Pay-As-You-Go	Create Time:
Instance Specification api.s1.small	Max Request Per Second: SLA: Max Connections: Max Internet Inbound Traffic: Max Internet Outbound Traffic:	2500 99.95% 50000 5120M 100M

4. In the left-side navigation pane, click API Groups.

5. Find the target API group and click the group name.

6. On the Group Details page, click Enable VPC Intranet Subdomain. You can also use a CNAME record to resolve your domain name to the VPC subdomain and then bind your domain name to the API group.


Group Details ↥ Back to group list Refresh

Basic Information | Turn on cloud monitoring | Api List | Modify Group Message

Region: China North 2 (Beijing)	Group Name: share	Group ID: [redacted]	
Subdomain Name	Internet Subdomain: [redacted]oudapi.com (The subdomain is only for API test, when the client directly calls it, there will be 1000 access restrictions per day. It is recommended to use the independent domain name for group binding, and it will not be subject to this restriction. For details, see configuration process) API gateway self-calling domain name: Disable (This function is for the dedicated instance use.) VPC Intranet Subdomain: Disable		Disable Internet Subdomain Enable VPC Intranet Subdomain
Instance Type: Classic Shared	Group Traffic Limit (QPS): 500 (If you need more, please buy a dedicated instance.)	Modify API Group's Instance	Instance Type And Selection Guide
Network Access Policy	HTTPS Security Policy: HTTPS2_TLS1_0	Change HTTPS Security Policy	HTTPS Security Policy Documentation
Legal Status: NORMAL			
Description:			

Usage notes:

- If no VPC is configured for the dedicated instance, VPC subdomains cannot be enabled for API groups under the instance.
- If the VPC configured for the dedicated instance has changed, VPC subdomains for all API groups under the instance are disabled.
- If an API group that is configured with a VPC subdomain is to be migrated from a shared instance (classic network or VPC) to a dedicated instance, you must configure a VPC for the dedicated instance before you can migrate the instance. **Note that you can access APIs only through the configured VPC.**

8.Access API Gateway from Function Compute over an internal network

This topic describes how to access API Gateway from Function Compute over a VPC.

1. Overview

API Gateway can be integrated with Function Compute to build a serverless architecture. You may need to call APIs published on API Gateway when you use Function Compute. For security purposes, we recommend that you access API Gateway over an internal network when you use Function Compute. This topic focuses on access to API Gateway over an internal network in the following scenarios:

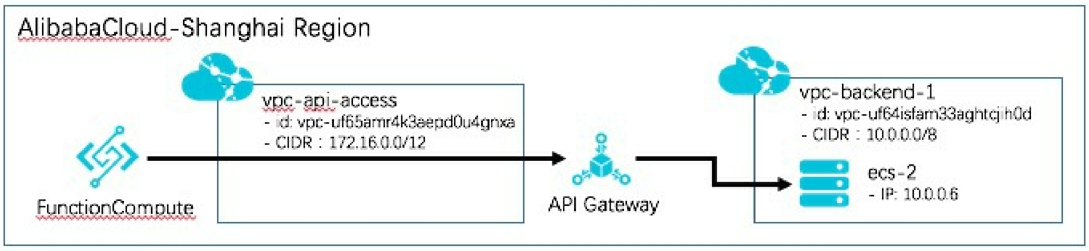
- Scenario 1: API Gateway and Function Compute are deployed in the same region.
- Scenario 2: API Gateway and Function Compute are deployed in different regions.

Follow these configuration principles in the two scenarios:

- Configure a VPC to access API Gateway. For more information, see Access API Gateway over a VPC.
- Configure a VPC to access Function Compute. For more information, see VPC Access

2. Scenario 1

Step 1: Build an architecture similar to the following figure.



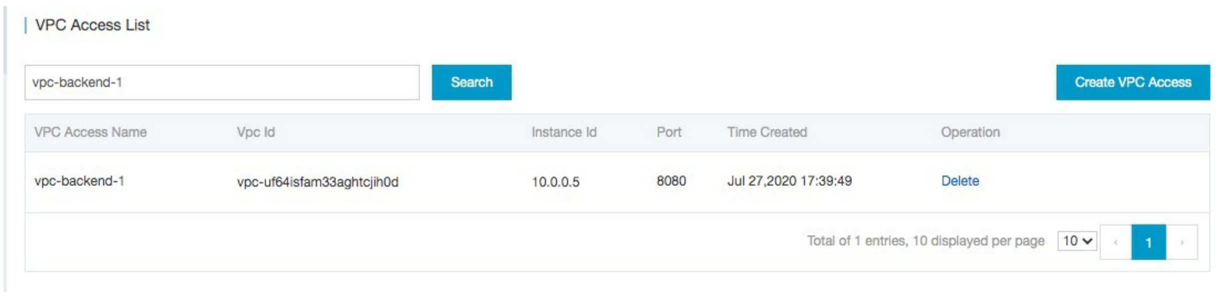
Note:

- Two VPCs named vpc-api-access and vpc-backend-1 are created in the China (Shanghai) region.
- One dedicated API Gateway instance is created in the China (Shanghai) region.
- In vpc-backend-1, an Elastic Compute Service (ECS) instance is created as the backend service of API Gateway. This ECS instance provides an API over HTTP. The HTTP service address is http://localhost:8080/web/cloudapi. A security group is configured to allow API Gateway to access this ECS instance.

Step 2: Create an API, with VPC as the backend service type.

For more information, see Use a resource in a VPC as the backend service of an API operation Take note of the following points:

- Create a VPC authorization. The created VPC authorization is displayed on the VPC Access List page.



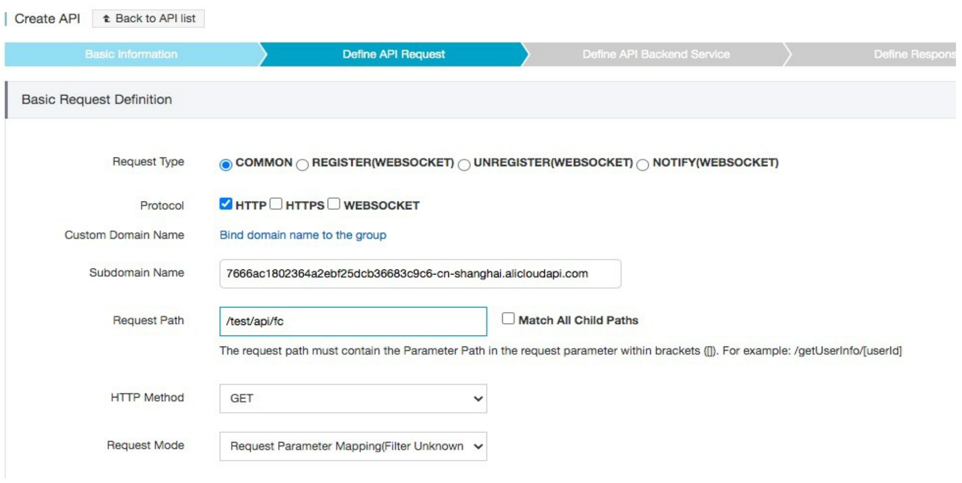
VPC Access List

vpc-backend-1 | Search | Create VPC Access

VPC Access Name	Vpc Id	Instance Id	Port	Time Created	Operation
vpc-backend-1	vpc-uf64isfam33aghtcjih0d	10.0.0.5	8080	Jul 27,2020 17:39:49	Delete

Total of 1 entries, 10 displayed per page 10 1

- Set Backend Service Type to VPC. Set Security Certification to No Certification to facilitate subsequent API call testing.

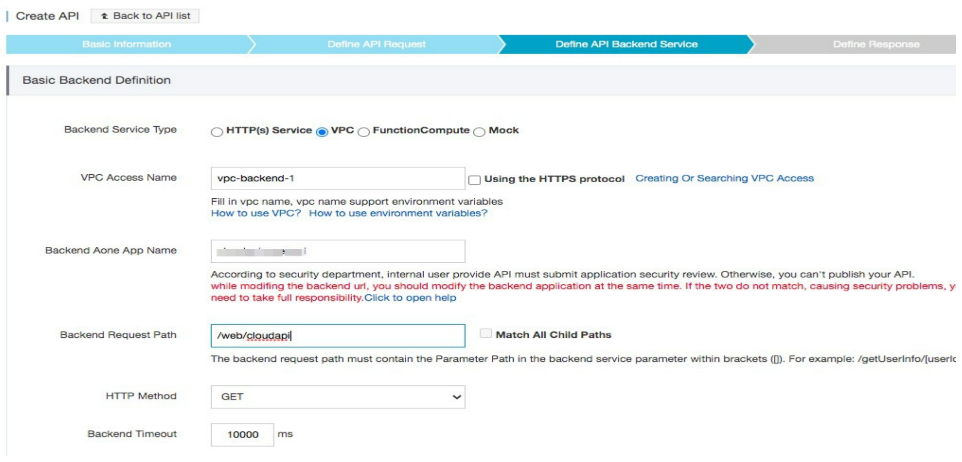


Create API | Back to API list

Basic Information > Define API Request > Define API Backend Service > Define Respons

Basic Request Definition

Request Type	[✓ COMMON] REGISTER(WEBSOCKET) UNREGISTER(WEBSOCKET) NOTIFY(WEBSOCKET)
Protocol	[✓ HTTP] HTTPS WEBSOCKET
Custom Domain Name	Bind domain name to the group
Subdomain Name	7666ac1802364a2ebf25dcb36683c9c6-cn-shanghai.alicloudapi.com
Request Path	/test/api/fc ☐ Match All Child Paths The request path must contain the Parameter Path in the request parameter within brackets ([]). For example: /getUserInfo/[userId]
HTTP Method	GET
Request Mode	Request Parameter Mapping(Filter Unknown



Create API | Back to API list

Basic Information > Define API Request > Define API Backend Service > Define Response

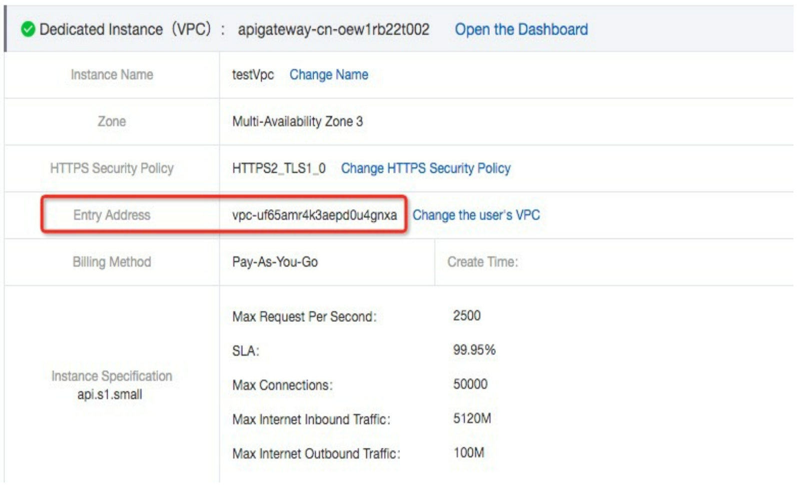
Basic Backend Definition

Backend Service Type	HTTP(s) Service [✓ VPC] FunctionCompute Mock
VPC Access Name	vpc-backend-1 ☐ Using the HTTPS protocol Creating Or Searching VPC Access Fill in vpc name, vpc name support environment variables How to use VPC? How to use environment variables?
Backend Aone App Name	[redacted] According to security department, internal user provide API must submit application security review. Otherwise, you can't publish your API. while modifing the backend url, you should modify the backend application at the same time. If the two do not match, causing security problems, y need to take full responsibility.Click to open help
Backend Request Path	/web/cloudapi ☐ Match All Child Paths The backend request path must contain the Parameter Path in the backend service parameter within brackets ([]). For example: /getUserInfo/[userIc
HTTP Method	GET
Backend Timeout	10000 ms

- Save the settings and publish the API to the production environment.

Step 3: Configure permissions on access to API Gateway over a VPC.

In the Dedicated Instance (VPC) section of the Instance list page in the API Gateway console, click Bind to the user's VPC on the right of the Entry Address field. In the Bind the VPC network of the dedicated instance dialog box, select the ID of vpc-api-access, and click OK. You can access API Gateway over vpc-api-access.

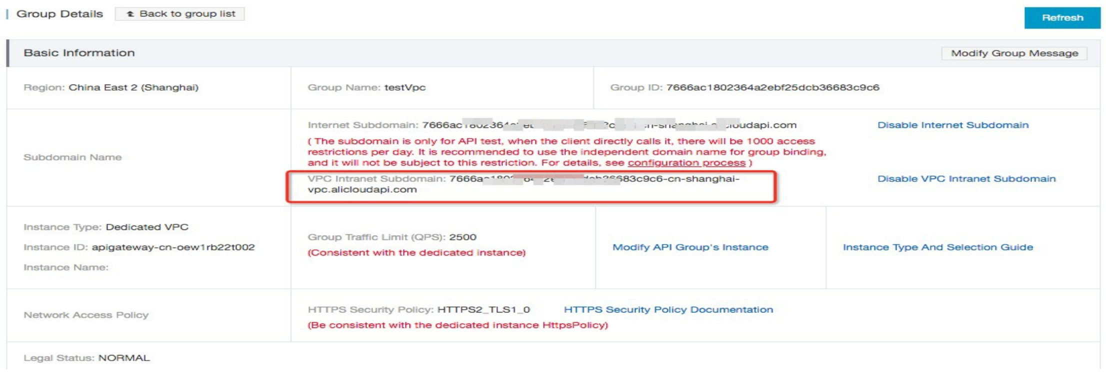


Dedicated Instance (VPC) : apigateway-cn-oew1rb22t002 Open the Dashboard

Instance Name	testVpc Change Name
Zone	Multi-Availability Zone 3
HTTPS Security Policy	HTTPS2_TLS1_0 Change HTTPS Security Policy
Entry Address	vpc-uf65amr4k3aepd0u4gnxa Change the user's VPC
Billing Method	Pay-As-You-Go — Create Time:
Instance Specification api.s1.small	Max Request Per Second: 2500 SLA: 99.95% Max Connections: 50000 Max Internet Inbound Traffic: 5120M Max Internet Outbound Traffic: 100M

Step 4: Enable a VPC subdomain for an API group.

On the Group Details page, click Enable VPC Intranet Subdomain on the right of the Subdomain Name field. In the Enable VPC Intranet Subdomain message, click OK. A VPC subdomain is assigned to this API group. You can use this subdomain to call APIs in this API group.



Group Details Back to group list Refresh

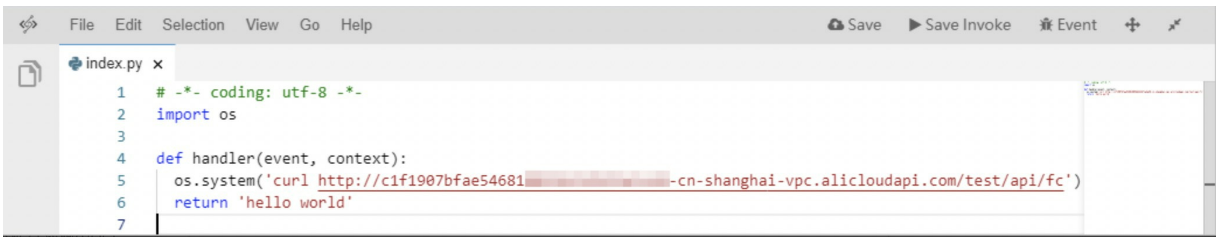
Basic Information Modify Group Message

Region: China East 2 (Shanghai)	Group Name: testVpc	Group ID: 7666ac1802364a2ebf25dcb36683c9c6
Subdomain Name	Internet Subdomain: 7666ac[redacted]cn-shanghai[redacted]cloudapi.com (The subdomain is only for API test, when the client directly calls it, there will be 1000 access restrictions per day. It is recommended to use the independent domain name for group binding, and it will not be subject to this restriction. For details, see configuration process) VPC Intranet Subdomain: 7666a[redacted]683c9c6-cn-shanghai-vpc.alicloudapi.com	Disable Internet Subdomain Disable VPC Intranet Subdomain
Instance Type: Dedicated VPC Instance ID: apigateway-cn-oew1rb22t002 Instance Name:	Group Traffic Limit (QPS): 2500 (Consistent with the dedicated instance)	Modify API Group's Instance — Instance Type And Selection Guide
Network Access Policy	HTTPS Security Policy: HTTPS2_TLS1_0 HTTPS Security Policy Documentation (Be consistent with the dedicated instance HttpsPolicy)	
Legal Status: NORMAL		

Note: Internet access is enabled for the API group by default. You can click Disable Internet Subdomain to disable Internet access based on your business requirements. If Internet access is disabled, you cannot perform online debugging in the API Gateway console.

Step 5: Create a function.

In Function Compute, create an application and create a function that runs in the Python environment. The following figure shows the content of this function.

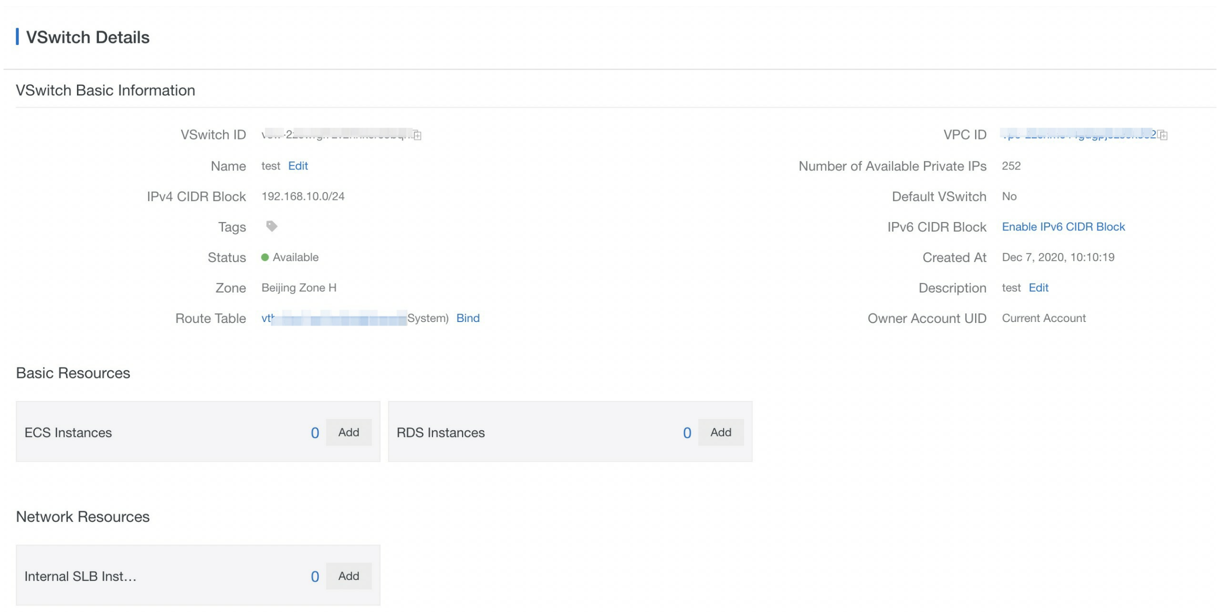


```python
# -*- coding: utf-8 -*-
import os

def handler(event, context):
  os.system('curl http://c1f1907bfae54681[redacted]-cn-shanghai-vpc.alicloudapi.com/test/api/fc')
  return 'hello world'

```

In this function, only the curl command is run to call APIs under a VPC subdomain. If you run the curl command to call APIs bound to the VPC subdomain, the API call fails.

Step 6: Configure VPC access for Function Compute.

Create a VSwitch in vpc-api-access to connect to Function Compute.



VSwitch Details

VSwitch Basic Information

VSwitch ID	[illegible]	VPC ID	[illegible]
Name	test Edit	Number of Available Private IPs	252
IPv4 CIDR Block	192.168.10.0/24	Default VSwitch	No
Tags		IPv6 CIDR Block	Enable IPv6 CIDR Block
Status	Available	Created At	Dec 7, 2020, 10:10:19
Zone	Beijing Zone H	Description	test Edit
Route Table	vt[illegible]System) Bind	Owner Account UID	Current Account

Basic Resources

ECS Instances 0 Add

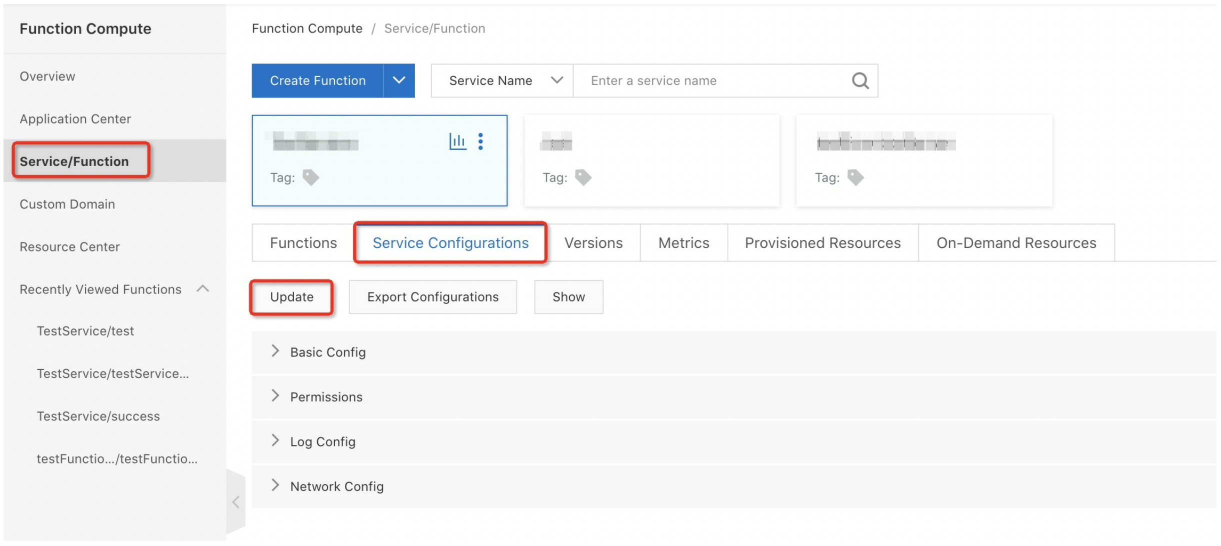
RDS Instances 0 Add

Network Resources

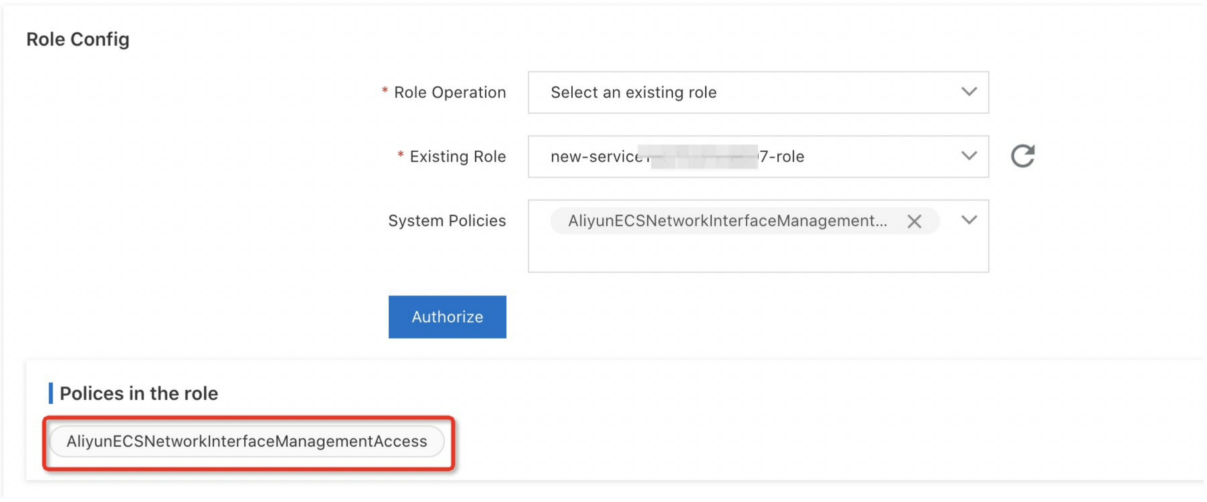
Internal SLB Inst... 0 Add

Log on to the Function Compute console and perform the following configurations. For more information, see VPC Access

In the Network Config section of the Configure Service page, select vpc-api-access from the VPC drop-down list and select the created VSwitch from the Vswitches drop-down list.



In the Role Config section of the Configure Service page, select Create new role from the Role Operation drop-down list, select AliyunECSNetworkInterfaceManagementAccess from the System Policies drop-down list, click Authorize, and then click Submit.



Role Config

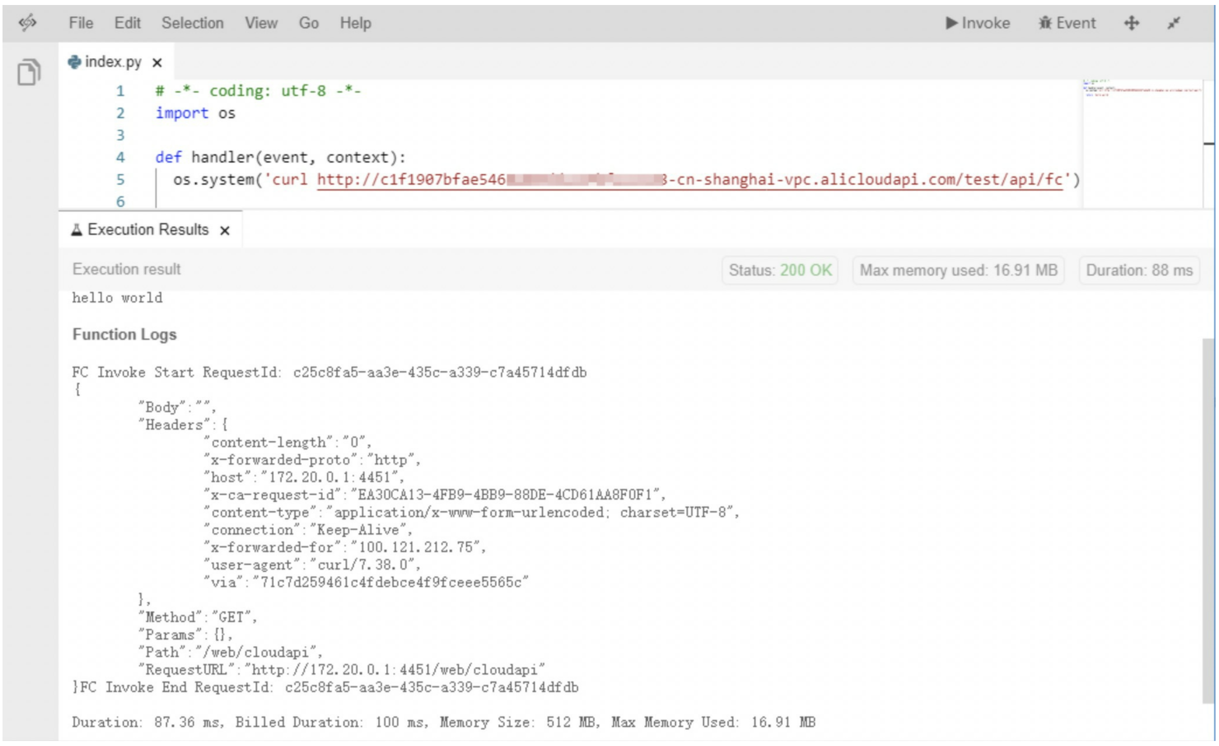
* Role Operation	Select an existing role
* Existing Role	new-service[redacted]7-role
System Policies	AliyunECSNetworkInterfaceManagement...

Polices in the role

AliyunECSNetworkInterfaceManagementAccess

Step 7: Execute the function for a test.

After you execute this function, you can use this VPC subdomain to access APIs published on API Gateway.



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

def handler(event, context):
  os.system('curl http://c1f1907bfae546[redacted]3-cn-shanghai-vpc.alicloudapi.com/test/api/fc')
```

Execution result Status: 200 OK Max memory used: 16.91 MB Duration: 88 ms

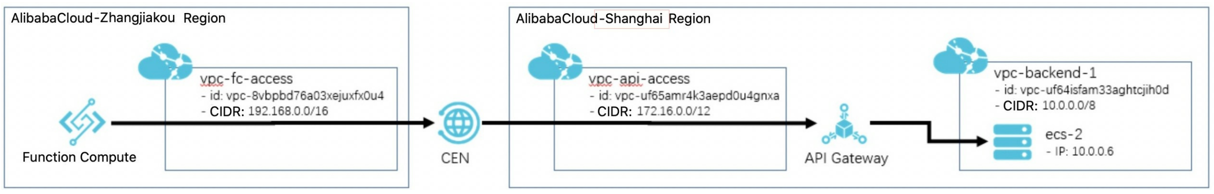
```
hello world
```

Function Logs

```
FC Invoke Start RequestId: c25c8fa5-aa3e-435c-a339-c7a45714dfdb
{
        "Body":"",
        "Headers":{
                "content-length":"0",
                "x-forwarded-proto":"http",
                "host":"172.20.0.1:4451",
                "x-ca-request-id":"EA30CA13-4FB9-4BB9-88DE-4CD61AA8F0F1",
                "content-type":"application/x-www-form-urlencoded; charset=UTF-8",
                "connection":"Keep-Alive",
                "x-forwarded-for":"100.121.212.75",
                "user-agent":"curl/7.38.0",
                "via":"71c7d259461c4fdebce4f9fceee5565c"
        },
        "Method":"GET",
        "Params":{},
        "Path":"/web/cloudapi",
        "RequestURL":"http://172.20.0.1:4451/web/cloudapi"
}FC Invoke End RequestId: c25c8fa5-aa3e-435c-a339-c7a45714dfdb

Duration: 87.36 ms, Billed Duration: 100 ms, Memory Size: 512 MB, Max Memory Used: 16.91 MB
```

3. Scenario 2

Step 1: Build an architecture similar to the following figure.

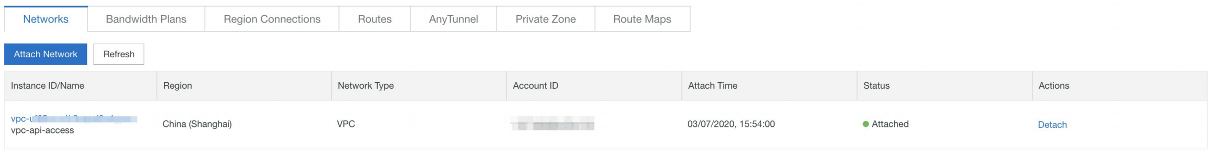

AlibabaCloud-Zhangjiakou Region
vpc-fc-access
- id: vpc-8vbpbd76a03xejuxfx0u4
- CIDR: 192.168.0.0/16
Function Compute
CEN
AlibabaCloud-Shanghai Region
vpc-api-access
- id: vpc-uf65amr4k3aepd0u4gnxa
- CIDR: 172.16.0.0/12
API Gateway
vpc-backend-1
- id: vpc-uf64isfam33aghtcjih0d
- CIDR: 10.0.0.0/8
ecs-2
- IP: 10.0.0.6

Note:

- In the China (Zhangjiakou-Beijing Winter Olympics) region, an application is created for Function Compute to initiate API calls. A VPC named vpc-fc-access is created. Function Compute can connect to the Cloud Enterprise Network (CEN) over this VPC.
- In the China (Shanghai) region, a dedicated API Gateway instance and a VPC named vpc-api-access are connected. API Gateway can connect to the CEN over this VPC.
- In the China (Shanghai) region, another VPC named vpc-backend-1 is created. An ECS instance is created as the backend service of API Gateway. This ECS instance provides an API over HTTP. The HTTP service address is http://localhost:8080/web/cloudapi. A security group is configured to allow API Gateway to access this ECS instance.

Step 2: Create a CEN.

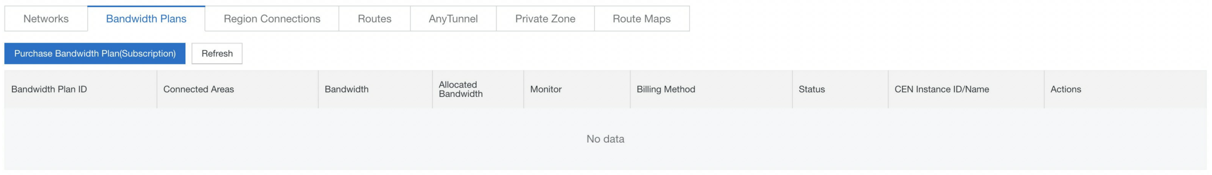
Create a CEN to interconnect vpc-api-access and vpc-fc-access, so Function Compute can communicate with API Gateway over an internal network. For more information about how to configure a CEN, see the CEN documentation. Log on to the CEN console and create a CEN instance. Associate vpc-api-access and vpc-fc-access with the CEN instance.



Instance ID/Name	Region	Network Type	Account ID	Attach Time	Status	Actions
[redacted] vpc-api-access	China (Shanghai)	VPC	[redacted]	03/07/2020, 15:54:00	● Attached	Detach

Step 3: Configure the bandwidth.

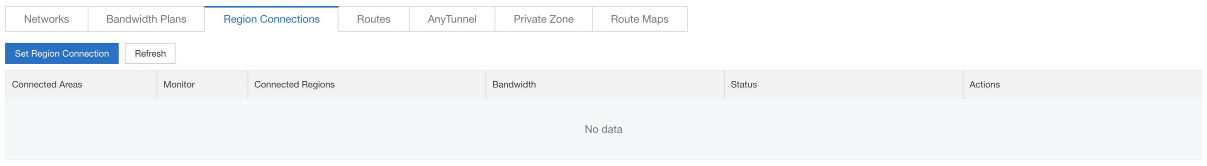
Purchase a bandwidth plan used for communication within the CEN instance. In this example, a bandwidth plan of 2 Mbit/s is purchased. You can purchase a bandwidth plan based on your business requirements.



Bandwidth Plan ID	Connected Areas	Bandwidth	Allocated Bandwidth	Monitor	Billing Method	Status	CEN Instance ID/Name	Actions

No data

On the Region Connections tab of the CEN page, configure the bandwidth for the regions of the CEN instance. You can configure the bandwidth size for each pair of connected regions based on the bandwidth plan.



Connected Areas	Monitor	Connected Regions	Bandwidth	Status	Actions

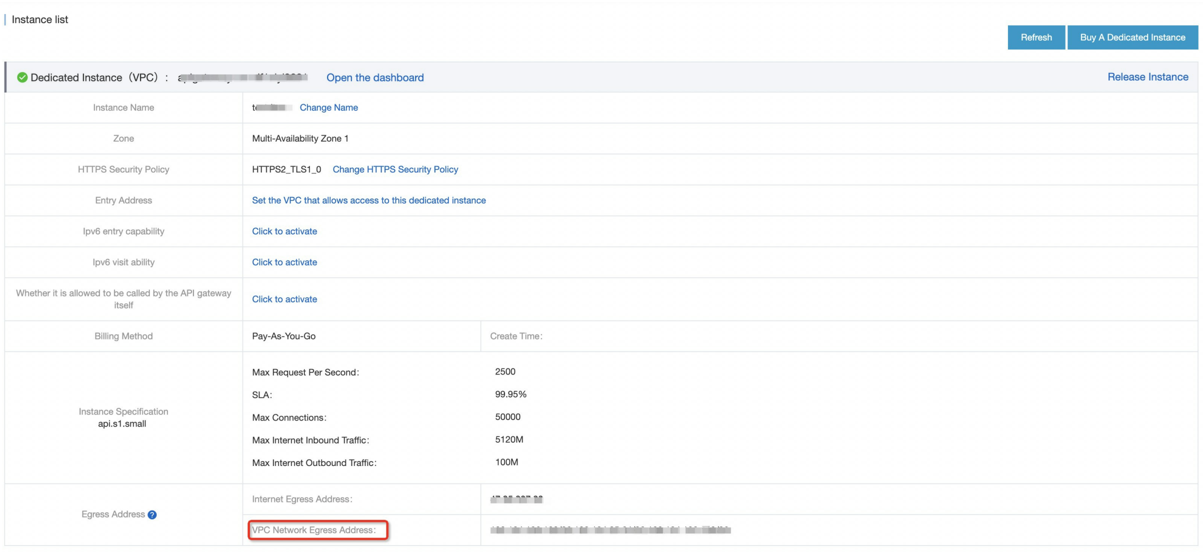
No data

Step 4: Configure cross-VPC routes.

Submit a ticket to the CEN product team to configure the routes. Provide the configuration parameters required by the Resolve And Route ServiceIn Cen API operation. These routes enable communication between API Gateway and the VPC in the China (Zhangjiakou-Beijing Winter Olympics) region.

```
AccessRegionIds.1=cn-zhangjiakou
AccessRegionIds.2=cn-shanghai
CenId=cen-uggzcthgz7cwsl7prr      #The ID of the CEN instance.
Host=100.104.255.128/26                #The VPC egress address of the dedicated API Gateway
instance.
HostRegionId=cn-shanghai
HostVpcId=vpc-uf65amr4k3aepd0u4gnxa     #The ID of vpc-api-access. API Gateway deployed in
the China (Shanghai) region can connect to the CEN over this VPC.
```

To obtain the VPC egress address of the dedicated API Gateway instance, go to the Instances page in the API Gateway console.



Dedicated Instance（VPC）： [redacted]

Instance Name	[redacted] Change Name	
Zone	Multi-Availability Zone 1	
HTTPS Security Policy	HTTPS2_TLS1_0 Change HTTPS Security Policy	
Entry Address	Set the VPC that allows access to this dedicated instance	
Ipv6 entry capability	Click to activate	
Ipv6 visit ability	Click to activate	
Whether it is allowed to be called by the API gateway itself	Click to activate	
Billing Method	Pay-As-You-Go	Create Time:
Instance Specification api.s1.small	Max Request Per Second:	2500
	SLA:	99.95%
	Max Connections:	50000
	Max Internet Inbound Traffic:	5120M
	Max Internet Outbound Traffic:	100M
Egress Address	Internet Egress Address:	[redacted]
	VPC Network Egress Address:	[redacted]

After your ticket is processed, you can view the routes configured for the China (Shanghai) and China (Zhangjiakou-Beijing Winter Olympics) regions in the CEN console. The custom routes are added based on the configuration parameters that you provided.

Step 5: Create an API, with VPC as the backend service type.

Follow the operations described in **Step 2** in **scenario 1**.

Step 6: Configure permissions on access to API Gateway over a VPC.

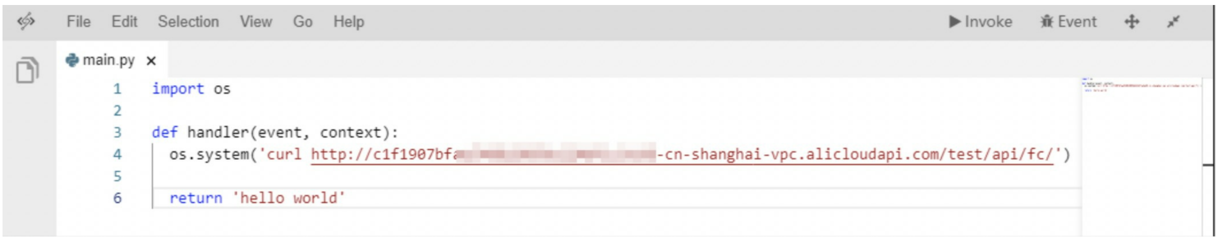
Follow the operations described in **Step 3** in **scenario 1**.

Step 7: Enable a domain name for an API group on the VPC.

Follow the operations described in **Step 4** in **scenario 1**.

Step 8: Create a function.

In the Function Compute instance deployed in the China (Zhangjiakou-Beijing Winter Olympics) region, create an application and create a function that runs in the Python environment. The following figure shows the content of this function.



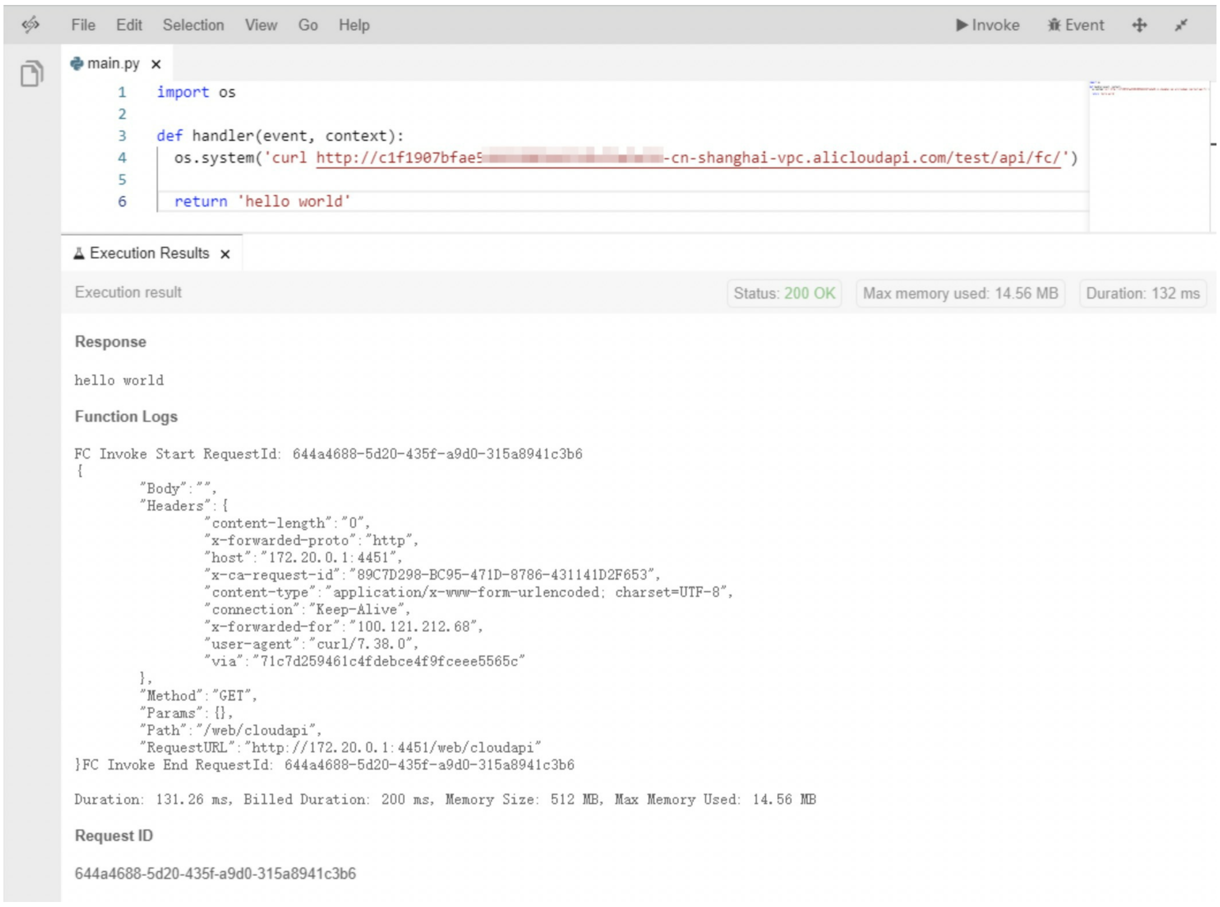
```python
import os

def handler(event, context):
  os.system('curl http://c1f1907bf[redacted]-cn-shanghai-vpc.alicloudapi.com/test/api/fc/')

  return 'hello world'
```

Step 9: Configure VPC access for Function Compute.

Create a VSwitch in vpc-fc-access. Then, log on to the Function Compute console and perform the required configurations. The operations are similar to those described in **Step 6** in **scenario 1**.

Step 10: Execute the function for a test.

After you execute this function, you can use this VPC subdomain to access APIs published on API Gateway.



```python
import os

def handler(event, context):
  os.system('curl http://c1f1907bfae5[redacted]-cn-shanghai-vpc.alicloudapi.com/test/api/fc/')

  return 'hello world'
```

Response

```
hello world
```

Function Logs

```
FC Invoke Start RequestId: 644a4688-5d20-435f-a9d0-315a8941c3b6
{
        "Body":"",
        "Headers":{
                "content-length":"0",
                "x-forwarded-proto":"http",
                "host":"172.20.0.1:4451",
                "x-ca-request-id":"89C7D298-BC95-471D-8786-431141D2F653",
                "content-type":"application/x-www-form-urlencoded; charset=UTF-8",
                "connection":"Keep-Alive",
                "x-forwarded-for":"100.121.212.68",
                "user-agent":"curl/7.38.0",
                "via":"71c7d259461c4fdebce4f9fceee5565c"
        },
        "Method":"GET",
        "Params":{},
        "Path":"/web/cloudapi",
        "RequestURL":"http://172.20.0.1:4451/web/cloudapi"
}FC Invoke End RequestId: 644a4688-5d20-435f-a9d0-315a8941c3b6

Duration: 131.26 ms, Billed Duration: 200 ms, Memory Size: 512 MB, Max Memory Used: 14.56 MB
```

Request ID

644a4688-5d20-435f-a9d0-315a8941c3b6

4. Limits

- You can only use a dedicated API Gateway instance.

9.Understand how an SDK implements bidirectional communication

API Gat eway provides t he bidirect ional communicat ion capabilit y. However, t he SDKs t hat are aut omat ically generat ed by API Gat eway are limit ed t o a f ew coding languages. If you need SDKs f or ot her coding languages, you must develop t hem yourself . This t opic describes in det ail t he SDKs t hat are aut omat ically generat ed by API Gat eway f or bidirect ional communicat ion. This helps you develop SDKs f or ot her coding languages. API Gat eway provides SDKs f or Android, Object ive-C, and Java t o support bidirect ional communicat ion. You can develop your own SDK in anot her coding language, as inst ruct ed in t his t opic, based on t he SDKs t hat are provided by API Gat eway.

For inf ormat ion about how t o use t he bidirect ional communicat ion f eat ure of API Gat eway, see Implement bidirectional communication .

1. Communications protocol and channel command words

In bidirect ional communicat ion, a client communicat es wit h API Gat eway by using a WebSocket channel. The WebSocket channel support s t wo communicat ions met hods: calling API operat ions and sending commands. This sect ion describes t he communicat ions rules and channel command words t hat are conf igured f or t he WebSocket channel.

1.1 Format conversion of API requests

In bidirect ional communicat ion, each API request or response must be t ransmit t ed as a set of JSON st rings. The WebSocket channel bet ween a client and API Gat eway convert s each HTTP request t o a set of JSON st rings and sends t he JSON st rings t o API Gat eway. The f ollowing code snippet shows t he conversion logic:

```
public class WebSocketApiRequest {
    String method;
    String host;
    String path;
    Map<String , String> querys;
    Map<String, List<String>> headers;
    int isBase64 = 0;
    String body;
}
```

For example, t he f ollowing code snippet shows a st andard HTTP request :

```
POST /http2test/test? param1=test HTTP/1.1
host: apihangzhou444.foundai.com
x-ca-seq:0
x-ca-key:12344133
ca_version:1
content-type:application/x-www-form-urlencoded; charset=utf-8
x-ca-timestamp:1525872629832
date:Wed, 09 May 2018 13:30:29 GMT+00:00
x-ca-signature:kYjdIuCnCrxx+EyLMTTx5NiXxqvfTTHBQAe68tv33Tw=
user-agent:ALIYUN-ANDROID-DEMO
x-ca-nonce:c9f15cbf-f4ac-4a6c-b54d-f51abf4b5b44
x-ca-signature-headers:x-ca-seq,x-ca-key,x-ca-timestamp,x-ca-nonce
content-length:33

username=xiaoming&password=123456789
```

The WebSocket protocol converts the HTTP request to a set of JSON strings, as shown in the following code snippet, and then sends the JSON strings to API Gateway.

```
{
 "headers": {
  "accept": ["application/json; charset=utf-8"],
  "host": ["apihangzhou444.foundai.com"],
  "x-ca-seq": ["0"],
  "x-ca-key": ["12344133"],
  "ca_version": ["1"],
  "content-type": ["application/x-www-form-urlencoded; charset=utf-8"],
  "x-ca-timestamp": ["1525872629832"],
  "date": ["Wed, 09 May 2018 13:30:29 GMT+00:00"],
  "x-ca-signature": ["kYjdIuCnCrxx+EyLMTTx5NiXxqvfTTHBQAe68tv33Tw="],
  "user-agent": ["ALIYUN-ANDROID-DEMO"],
  "x-ca-nonce": ["c9f15cbf-f4ac-4a6c-b54d-f51abf4b5b44"],
  "x-ca-signature-headers": ["x-ca-seq,x-ca-key,x-ca-timestamp,x-ca-nonce"]
 },
 "host": "apihangzhou444.foundai.com",
 "isBase64": 0,
 "method": "POST",
 "path": "/http2test/test",
 "querys": {"param1":"test"},
 "body":"username=xiaoming&password=123456789"
}
```

1.2 Channel commands

In bidirectional communication, apart from calling API operations, a client can also send channel commands. This section describes the channel commands that are supported between a client and API Gateway.

1.2.1 Commands related to client registration

```
Command word: RG
Description: establishes a persistent connection between a client and API Gateway. This com
mand must include the device ID of the client.
Command type: request
Sender: client
Format: RG#DeviceId
Example: RG#ffd3234343dae324342@12344133

Command word: RO
Description: returns a success response to indicate that the persistent connection has been
established and API Gateway has received the device ID of the client. This command must inc
lude the unique identifier of the persistent connection and the required heartbeat interval
.
Command type: response
Sender: API Gateway
Format: RO#ConnectionCredential#keepAliveInterval
Example: RO#1534692949977#25000

Command word: RF
Description: returns an error response to indicate that the connection failed to be establi
shed.
Command type: response
Sender: API Gateway
Format: RF#ErrorMessage
Example: RF#ServerError
```

1.2.2 Commands related to a client's heartbeat

```
Command word: H1
Description: sends a heartbeat signal to API Gateway.
Command type: request
Sender: client
This command requires no parameters. You only need to send the command word.

Command word: HO
Description: returns a success response to indicate that API Gateway has received the heart
beat signal.
Command type: response
Sender: API Gateway
Format: HO#ConnectionCredential
Example: HO#ffd3234343dae324342
```

1.2.3 Commands related to server-to-client notifications

```
Command word: NF
Description: sends a request to push a server-to-client notification.
Command type: request
Sender: API Gateway
Format: NF#MESSAGE
Example: NF#HELLO WORLD!

Command word: NO
Description: returns a response to indicate that the client has received a server-to-client
notification.
Command type: response
Sender: client
This command requires no parameters. You only need to send the command word.
```

1.2.4 Other commands

```
Command word: OS
Description: If the number of API requests from a client reaches a threshold that is specif
ied by a throttling policy in API Gateway, API Gateway sends this command to the client. Th
e client must cut off the connection and register again. This process will not affect user
experience. If the client itself does not complete this process after it receives this comm
and, the connection will be cut off by API Gateway not long after.
Command type: request
Sender: API Gateway
This command requires no parameters. You only need to send the command word.

Command word: CR
Description: If the persistent connection between a client and API Gateway reaches the end
of its lifecycle, API Gateway sends this command to the client. The client must cut off the
connection and register again. This process will not affect user experience. If the client
itself does not complete this process after it receives this command, the connection will b
e cut off by API Gateway not long after.
Command type: request
Sender: API Gateway
This command requires no parameters. You only need to send the command word.
```

1.2.5 Unique identifier of a device

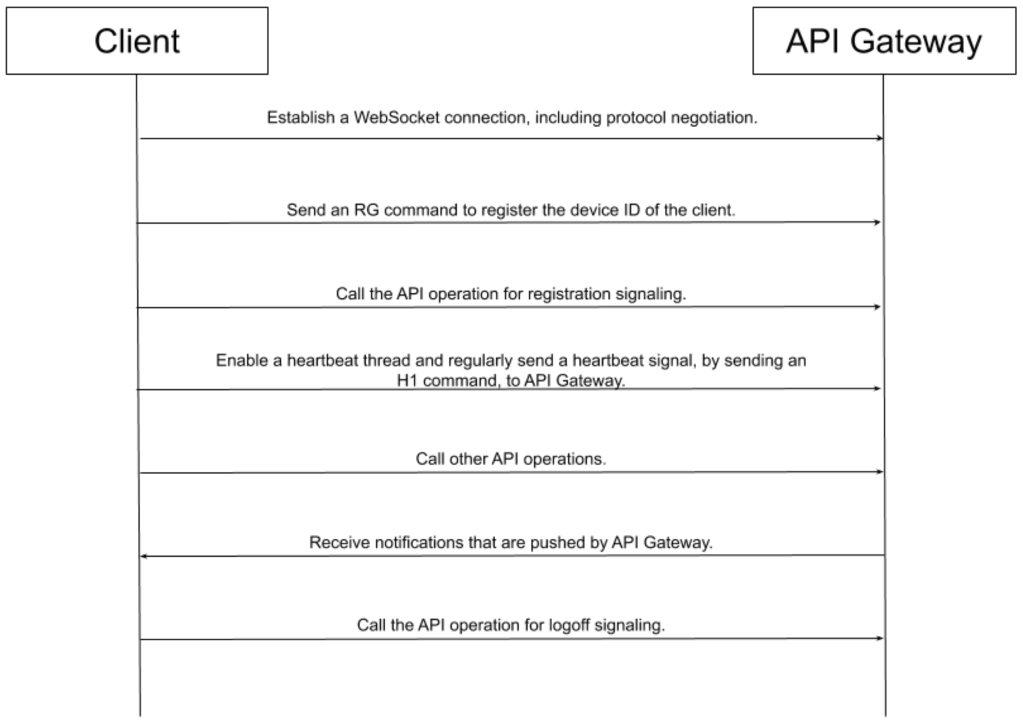
The device ID of a client is used to uniquely identify the persistent connection between the client and API Gateway. The format of a device ID must be UUID@AppKey. The following code snippet shows how to use Java code to specify a device ID. For the same application, each device ID must be unique among all the clients that use the application. Namely, for the same application, all device IDs must be different from each other. If two clients have the same device ID, an error will occur during registration.

```
String deviceId = UUID.randomUUID().toString().replace("-" , "")+ "@" + appKey;
```

For example, a device ID can be ffd3234343dae324342@12344133.

2. Communications process

The following figure shows the interaction between API Gateway and an SDK that is installed on a client.


Client
API Gateway
Establish a WebSocket connection, including protocol negotiation.
Send an RG command to register the device ID of the client.
Call the API operation for registration signaling.
Enable a heartbeat thread and regularly send a heartbeat signal, by sending an H1 command, to API Gateway.
Call other API operations.
Receive notifications that are pushed by API Gateway.
Call the API operation for logoff signaling.

As shown in the figure, the SDK on the client is required to complete the following operations:

- Establish a WebSocket connection, including protocol negotiation.
- Send an RG command to register the device ID of the client.
- Call the API operation for registration signaling.
- Enable a heartbeat thread and regularly send a heartbeat signal, by sending an H1 command, to API Gateway.
- Call other API operations.
- Receive notifications that are pushed by API Gateway.
- Call the API operation for logoff signaling.

The SDK on the client is also required to handle the following exceptions:

- After the persistent connection between the client and API Gateway reaches the end of its lifecycle, the SDK needs to cut off the connection and register again.
- After the number of API requests from the client reaches a threshold that is specified by a throttling policy in API Gateway, the SDK needs to cut off the connection and register again.

2.1 Establish a WebSocket connection, including protocol negotiation

API Gateway implements bidirectional communication based on the WebSocket protocol. Each connection between a client and API Gateway is a standard WebSocket connection. For information about how to establish a WebSocket connection, see The WebSocket Protocol.

The WebSocket protocol can be implemented by using common coding languages. For example, Android applications use OkHttp to implement the WebSocket protocol.

2.2 Send an RG command to register the device ID of the client

After the WebSocket connection is established, the client sends a message to API Gateway to inform API Gateway of the device ID of the client.

After API Gateway receives the message, API Gateway establishes a persistent connection at the access layer and uses the device ID of the client to identify this persistent connection. Then, API Gateway returns a success response to the client, indicating that the connection has been established.

For information about the commands that are involved in this process, see section 1.2.1.

2.3 Call the API operation for registration signaling

After the WebSocket connection is established, the client sends a registration signal, namely, calls the API operation for registration signaling. You must have created the API operation for registration signaling in API Gateway in advance. For information about how to create the API operation for registration signaling, see Part three of Implement bidirectional communication.

For information about the format of an API request, see section 1.1. When you send a request for the API operation for registration signaling, take note of the following items:

1. Send the request only after API Gateway sends an RO command in response to the RG command from the client.
2. Add the following header field in the request to identify the API operation for registration signaling: x-ca-websocket_api_type:REGISTER.

2.4 Enable a heartbeat thread and regularly send a heartbeat signal, by sending an H1 command, to API Gateway

The client is required to send a heartbeat signal to API Gateway every 25 seconds. Each time API Gateway receives a heartbeat signal, API Gateway updates the lifespan of the connection between the client and API Gateway, and then returns a success response to indicate that the heartbeat signal has been received and handled.

For information about the commands that are involved in this process, see section 1.2.2.

2.5 Call other API operations

After the WebSocket connection is established, the client can also call other API operations that are unrelated to bidirectional communication. For information about the format of an API request, see section 1.1.

2.6 Receive notifications that are pushed by API Gateway

After the client sends a request for the API operation for registration signaling and receives a 200 response from API Gateway, the client is ready to receive server-to-client notifications from API Gateway. For information about the commands that are related to sending and receiving server-to-client notifications, see section 1.2.3.

2.7 Call the API operation for logoff signaling

To log off from the application, the client needs to send a request for the API operation for logoff signaling. You must have created the API operation for logoff signaling in API Gateway in advance. For information about how to create the API operation for registration signaling, see Part three of Implement bidirectional communication.

For information about the format of an API request, see section 1.1. When you send a request for the API operation for logoff signaling, add the following header field in the request to identify the API operation for logoff signaling: x-ca-websocket_api_type:UNREGISTER.

2.8 Handle a CR command that is sent from API Gateway because of connection expiration

Each persistent connection has a lifecycle. The lifecycle of a persistent connection to API Gateway ends after 2,000 API requests are sent by using this connection. When the number of API requests that are sent by using a persistent connection reaches 1,500, API Gateway sends a CR command to the client, which requires the client to cut off the connection and register again. For more information about the CR command, see section 1.2.4. If the client does not cut off the connection and register again, API Gateway will delete the connection when the number of API requests that are sent by using this connection reaches 2,000.

After a client receives a CR command or an OS command from API Gateway, the client stops receiving API requests and continues to send the API requests that are already received to API Gateway. Then, the client cuts off the current connection, establishes another connection with API Gateway, and then sends a registration signal to register again. We recommend that when you register a client for the first time, you cache the registration request. In this way, every time after you disconnect the client from API Gateway, you can use the cache to register again more conveniently.

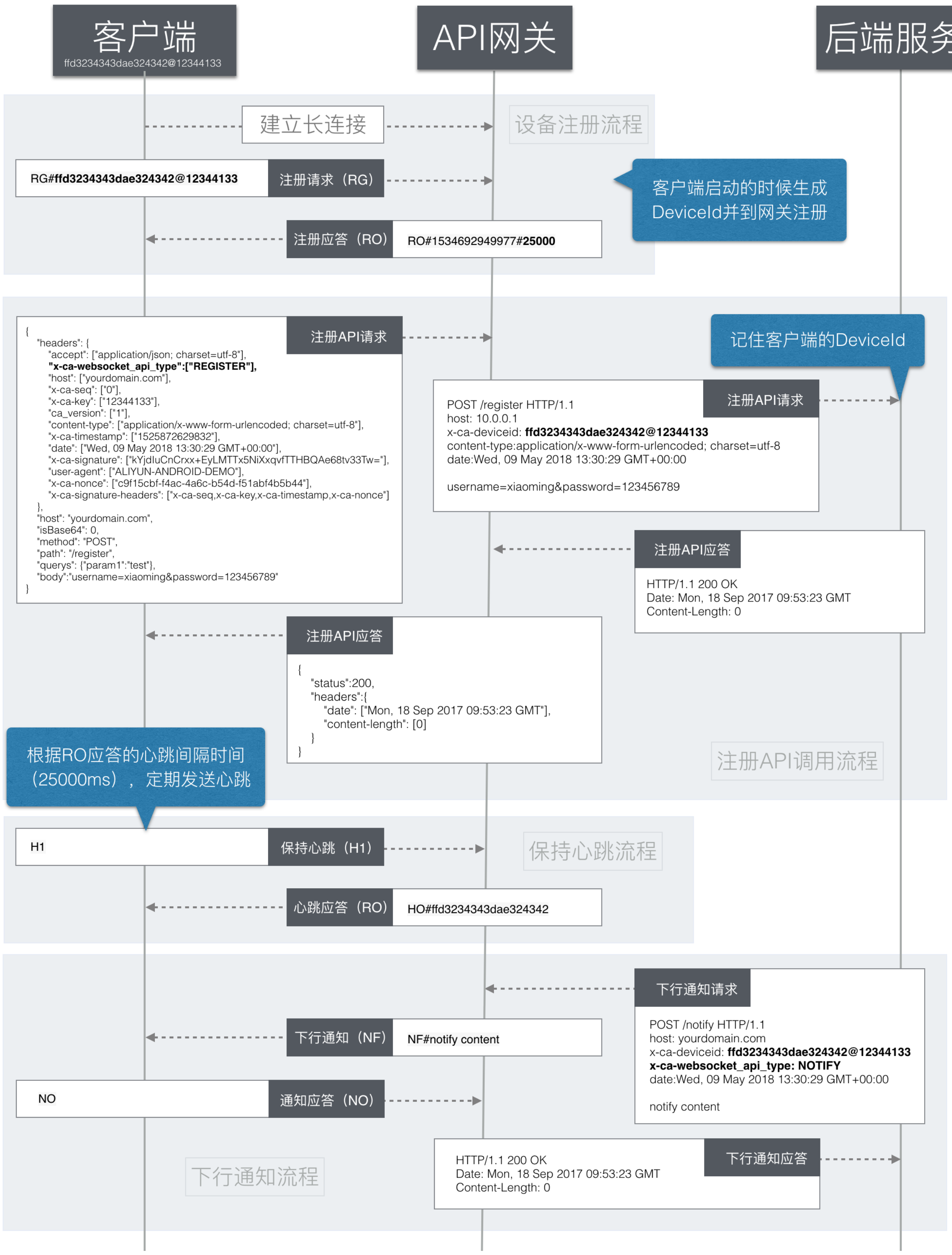
2.9 Handle an OS command that is sent by API Gateway because of throttling

API Gateway supports throttling. If the number of requests per second (RPS) on a client exceeds a threshold that is specified by a throttling policy, API Gateway sends an OS command to the client. For more information about the OS command, see section 1.2.4. To avoid receiving an OS command, the client must control the number of RPS. If the OS command is ignored and the number of RPS on the client continues to increase, API Gateway will delete the persistent connection between the client and API Gateway.

3. Summary

3.1 Process summary

The following figure shows the process of bidirectional communication in detail.



```
{
    "headers": {
        "accept": ["application/json; charset=utf-8"],
        "x-ca-websocket_api_type":["REGISTER"],
        "host": ["yourdomain.com"],
        "x-ca-seq": ["0"],
        "x-ca-key": ["12344133"],
        "ca_version": ["1"],
        "content-type": ["application/x-www-form-urlencoded; charset=utf-8"],
        "x-ca-timestamp": ["1525872629832"],
        "date": ["Wed, 09 May 2018 13:30:29 GMT+00:00"],
        "x-ca-signature": ["kYjdIuCnCrxx+EyLMTTx5NiXxqvfTTHBQAe68tv33Tw="],
        "user-agent": ["ALIYUN-ANDROID-DEMO"],
        "x-ca-nonce": ["c9f15cbf-f4ac-4a6c-b54d-f51abf4b5b44"],
        "x-ca-signature-headers": ["x-ca-seq,x-ca-key,x-ca-timestamp,x-ca-nonce"]
    },
    "host": "yourdomain.com",
    "isBase64": 0,
    "method": "POST",
    "path": "/register",
    "querys": {"param1":"test"},
    "body":"username=xiaoming&password=123456789"
}
```

```
POST /register HTTP/1.1
host: 10.0.0.1
x-ca-deviceid: ffd3234343dae324342@12344133
content-type:application/x-www-form-urlencoded; charset=utf-8
date:Wed, 09 May 2018 13:30:29 GMT+00:00

username=xiaoming&password=123456789
```

```
HTTP/1.1 200 OK
Date: Mon, 18 Sep 2017 09:53:23 GMT
Content-Length: 0
```

```
{
    "status":200,
    "headers":{
        "date": ["Mon, 18 Sep 2017 09:53:23 GMT"],
        "content-length": [0]
    }
}
```

```
POST /notify HTTP/1.1
host: yourdomain.com
x-ca-deviceid: ffd3234343dae324342@12344133
x-ca-websocket_api_type: NOTIFY
date:Wed, 09 May 2018 13:30:29 GMT+00:00

notify content
```

```
HTTP/1.1 200 OK
Date: Mon, 18 Sep 2017 09:53:23 GMT
Content-Length: 0
```

Note: The request format and response format of an API operation that is unrelated to bidirectional communication are the same as those of the API operation for registration signaling. Therefore, the figure does not provide examples of a request and a response of an API operation that is unrelated to bidirectional communication.

3.2 Sample code

API Gateway provides SDKs for Android, Objective-C, and Java to support the bidirectional communication capability. Each SDK is implemented strictly based on the communications rules and process that are described in the preceding sections. This section uses the SDK for Android as an example and provides sample code for part of the process in section 2.

3.2.1 Establish a WebSocket connection, including protocol negotiation

```
// Use a domain name and port 8080 as the connection address. The domain name is an indepen
dent domain name that is bound to the API group to which the API operations for registratio
n signaling, server-to-client notification signaling, and logoff signaling belong.
String websocketUrl = "ws:" + yourdomain + ":8080";

// Create a client object.
OkHttpClient client = new OkHttpClient.Builder()
                .readTimeout(params.getReadTimeout(), TimeUnit.MILLISECONDS)
                .writeTimeout(params.getWriteTimeout(), TimeUnit.MILLISECONDS)
                .connectTimeout(params.getConnectionTimeout(), TimeUnit.MILLISECONDS)
                .build();

// Construct an HTTP request that is used to establish a WebSocket connection between the c
lient and API Gateway.
Request connectRequest = new Request.Builder().url(websocketUrl).build();

// Create a WebSocketListener class to be used to listen on server-to-client notifications
after the WebSocket connection is established.
webSocketListener = new WebSocketListener() {
 ......
}

// Establish the persistent WebSocket connection.
client.newWebSocket(connectRequest, webSocketListener);
```

3.2.2 Send an RG command to register the device ID of the client or send a heartbeat signal to API Gateway

```
 String deviceId = generateDeviceId();
 webSocketListener = new WebSocketListener() {
                @Override
                // Call the onOpen method after the WebSocket connection is established.
                public void onOpen(WebSocket webSocket, Response response) {
                  // Send a message to API Gateway to inform API Gateway of the device ID o
f the client.
                    String registerCommand = SdkConstant.CLOUDAPI_COMMAND_REGISTER_REQUEST
+ "#" + deviceId;
                    webSocketRef.getObj().send(registerCommand);
                }

                @Override
                // Call the onMessage method after the client receives a response from API
Gateway.
                public void onMessage(WebSocket webSocket, String text) {
                    if(null == text || "".equalsIgnoreCase(text)) {
```

```
                    if(null  == text ||   .equalsIgnoreCase(text)) {
                        return;
                    }
                    // If the response from API Gateway is a success response, run the foll
owing code segment:
                    else if(text.length() > 2 && text.startsWith(SdkConstant.CLOUDAPI_COMMA
ND_REGISTER_SUCCESS_RESPONSE)){
                        // Parse the success response.
                        String responseObject[] = text.split("#");
                        connectionCredential = responseObject[1];
                        // Obtain the heartbeat interval in the success response.
                        heartBeatInterval = Integer.parseInt(responseObject[2]);

                        }

      // Enable a heartbeat thread to be used to send heartbeat signals.
                        if (null ! = heartBeatManager) {
                            heartBeatManager.stop();
                        }
                        heartBeatManager = new HeartBeatManager(instance, heartBeatInterval
);

                        heartbeatThread = new Thread(heartBeatManager);
                        heartbeatThread.start();
                        return;
                    }
                    // If the response from API Gateway is an error response, run the follo
wing code segment:
                    else if(text.length() > 2 && text.startsWith(SdkConstant.CLOUDAPI_COMMA
ND_REGISTER_FAIL_REQUEST)){

                        String responseObject[] = text.split("#");
                        errorMessage.setObj(responseObject[1]);
      // Stop the heartbeat thread.
                        if (null ! = heartBeatManager) {
                            heartBeatManager.stop();
                        }
                        return;
                }
            };
        }

    private String generateDeviceId(){
        return UUID.randomUUID().toString().replace("-" , "").substring(0 , 8);
    }
```

3.2.3 Call the API operation for registration signaling, the API operation for logoff signaling, and API operations that are unrelated to bidirectional communication

```
protected void sendAsyncRequest(final ApiRequest apiRequest , final ApiCallback apiCallback
){
        checkIsInit();
  // Check whether a connection has been established.
```

```
        synchronized (connectionLock) {
            if (null ! = connectLatch.getObj() && connectLatch.getObj().getCount() == 1) {
                try {
                    connectLatch.getObj().await(10, TimeUnit.SECONDS);
                } catch (InterruptedException ex) {
                    throw new SdkException("WebSocket connect server failed ", ex);
                } finally {
                    connectLatch.setObj(null);
                }
            }

            if (status == WebSocketConnectStatus.LOST_CONNECTION) {
                apiCallback.onFailure(apiRequest, new SdkException("WebSocket conection los
t , connecting"));
                return;
            }

   // If the API operation to be called is the API operation for registration signaling or
the API operation for logoff signaling, run the following code segment:
            if (WebSocketApiType.COMMON ! = apiRequest.getWebSocketApiType()) {
                if(! preSendWebsocketCommandApi(apiRequest , apiCallback)) {
                    return;
                }
            }

            Integer seqNumber = seq.getAndIncrement();
            apiRequest.addHeader(SdkConstant.CLOUDAPI_X_CA_SEQ, seqNumber.toString());
            callbackManager.add(seqNumber, new ApiContext(apiCallback, apiRequest));

            // Generate a request string to be sent.
            String request = buildRequest(apiRequest);
            webSocketRef.getObj().send(request);
        }

    }

    private boolean preSendWebsocketCommandApi(final ApiRequest apiRequest , final ApiCallb
ack apiCallback){
        // If the API operation to be called is the API operation for registration signalin
g, check whether API Gateway has been informed of the device ID of the client.
        if(WebSocketApiType.REGISTER == apiRequest.getWebSocketApiType()) {
            try {
                if (null ! = registerLatch.getObj() && ! registerLatch.getObj().await(10, T
imeUnit.SECONDS)) {
                    Thread.sleep(5000);
                    close();
                    apiCallback.onFailure(apiRequest, new SdkException("WebSocket conection
lost , connecting"));
                    return false;
                }
            } catch (InterruptedException ex) {
                throw new SdkException("WebSocket register failed ", ex);
            } finally {
                registerLatch.setObj(null);
            }

            if (! registerCommandSuccess.getObj()) {
                apiCallback.onFailure(null, new SdkException("Register Comand return error
:" + errorMessage.getObj()));
                return false;
            }

   // Record the registration request for future use in the event of disconnection and reco
nnection.
            lastRegisterReqeust = apiRequest.duplicate();
            lastRegisterCallback = apiCallback;

        }
  // If the API operation to be called is the API operation for registration signaling or t
he API operation for logoff signaling, add a header field in the API request to identify th
e API operation for registration signaling or the API operation for logoff signaling.
        apiRequest.addHeader(SdkConstant.CLOUDAPI_X_CA_WEBSOCKET_API_TYPE, apiRequest.getWe
bSocketApiType().toString());


        return true;
    }


     private String buildRequest(ApiRequest apiRequest){
        apiRequest.setHost(host);
        apiRequest.setScheme(scheme);
        ApiRequestMaker.make(apiRequest , appKey , appSecret);


        WebSocketApiRequest webSocketApiRequest = new WebSocketApiRequest();
        webSocketApiRequest.setHost(host);
        webSocketApiRequest.setPath(apiRequest.getPath());
        webSocketApiRequest.setMethod(apiRequest.getMethod().getValue());
        webSocketApiRequest.setQuerys(apiRequest.getQuerys());
        webSocketApiRequest.setHeaders(apiRequest.getHeaders());
        webSocketApiRequest.setIsBase64(apiRequest.isBase64BodyViaWebsocket() == true ? 1 :
0);
        MediaType bodyType = MediaType.parse(apiRequest.getFirstHeaderValue(HttpConstant.CL
OUDAPI_HTTP_HEADER_CONTENT_TYPE));

        if(null ! = apiRequest.getFormParams() && apiRequest.getFormParams().size() > 0){
            webSocketApiRequest.setBody(HttpCommonUtil.buildParamString(apiRequest.getFormP
arams()));
        }else if(null ! = apiRequest.getBody()){
            webSocketApiRequest.setBody(new String(apiRequest.getBody() , bodyType.charset(
SdkConstant.CLOUDAPI_ENCODING)));
        }

        if(apiRequest.isBase64BodyViaWebsocket()){
            webSocketApiRequest.setBody(new String(Base64.encode(apiRequest.getBody() , Bas
e64.DEFAULT) , bodyType.charset(SdkConstant.CLOUDAPI_ENCODING)));
        }
        return JSON.toJSONString(webSocketApiRequest);
}
```

3.2.4 Receive notifications that are pushed by API Gateway

```
 ApiWebSocketListner apiWebSocketListner = params.getApiWebSocketListner();
 webSocketListener = new WebSocketListener() {
  ......

                @Override
                // Call the onMessage method after the client receives a server-to-client n
otification from API Gateway.
                public void onMessage(WebSocket webSocket, String text) {
                     String message  = text.substring(3);
                     apiWebSocketListner.onNotify(message);
                     if(status == WebSocketConnectStatus.CONNECTED && webSocketRef.getObj()
! = null){
                            webSocketRef.getObj().send(SdkConstant.CLOUDAPI_COMMAND_NOTIFY_
RESPONSE);
                     }
                     return ;
              }

     ......
 }
```

For more information about sample code, download and decompress the SDK for Android. You can find more sample code in the src folder.

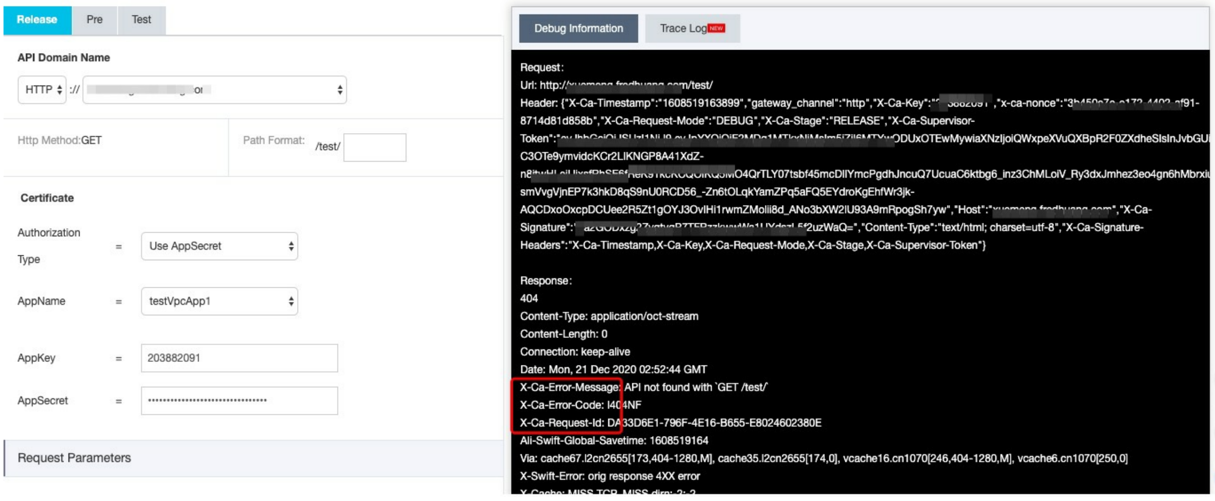
10.Troubleshooting

This topic describes how to identify and resolve issues that you may encounter when you debug APIs online.

1. Obtain error information

API Gateway returns a response to the request received from the client. You must check the response headers that start with X-Ca. Take note of the following points:

- - X-Ca-Request-Id: the unique ID of the request. When API Gateway receives a request, it generates a request ID and contains the request ID in the X-Ca-Request-Id header of the response message. We recommend that you record the request ID in both the client and your backend service for troubleshooting and tracking.
- - X-Ca-Error-Message: the error message returned by API Gateway. If a request fails, API Gateway contains the error message in the X-Ca-Error-Message header of the response message.
- - X-Ca-Error-Code: the error code of a system error in API Gateway. If a request is blocked by API Gateway due to an error, API Gateway contains the corresponding error code in the X-Ca-Error-Code header of the response message. Instances of the classic network type do not have this header.



API Domain Name
HTTP :// [redacted]

Http Method:GET
Path Format: /test/

Certificate

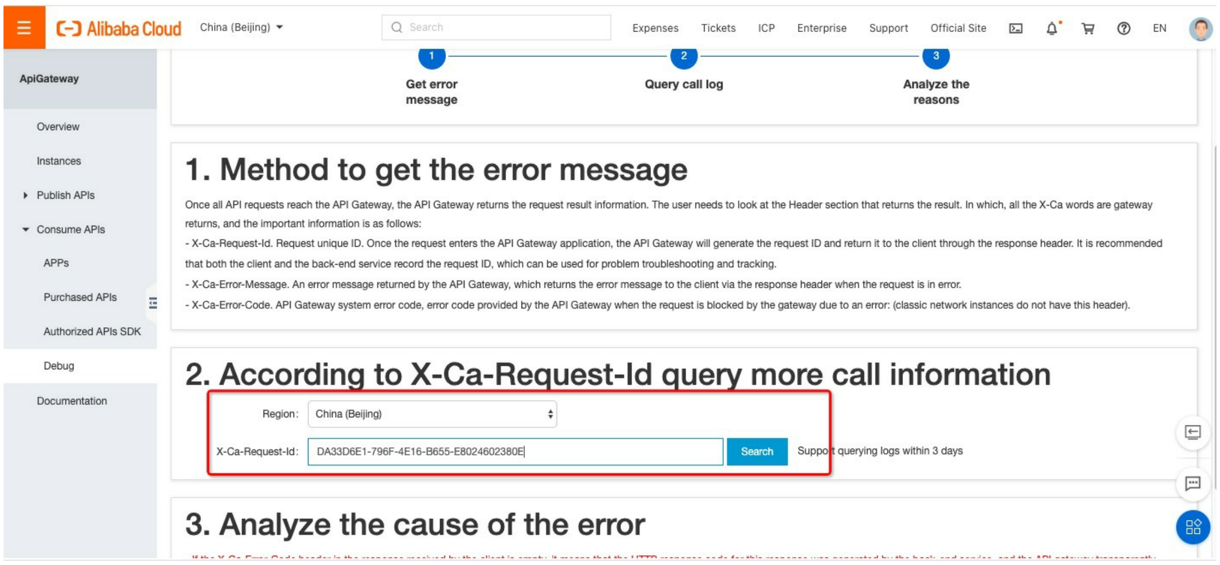
Authorization Type = Use AppSecret
AppName = testVpcApp1
AppKey = 203882091
AppSecret = ••••••••••••••••••••••••••••••••

```
Request:
Url: http://[redacted]m/test/
Header: {"X-Ca-Timestamp":"1608519163899","gateway_channel":"http","X-Ca-Key":"[redacted]","x-ca-nonce":"3[redacted]af91-
8714d81d858b","X-Ca-Request-Mode":"DEBUG","X-Ca-Stage":"RELEASE","X-Ca-Supervisor-
Token":"[redacted]ODUxOTEwMywiaXNzIjoiQWxpeXVuQXBpR2F0ZXdheSIsInJvbGU
C3OTe9ymvidcKCr2LIKNGP8A41XdZ-
n8[redacted]O4QrTLY07tsbf45mcDlIYmcPgdhJncuQ7UcuaC6ktbg6_inz3ChMLoiV_Ry3dxJmhez3eo4gn6hMbrxiu
smVvgVjnEP7k3hkD8qS9nU0RCD56_-Zn6tOLqkYamZPq5aFQ5EYdroKgEhfWr3jk-
AQCDxoOxcpDCUee2R5Zt1gOYJ3OvlHi1rwmZMolii8d_ANo3bXW2IU93A9mRpogSh7yw","Host":"[redacted]","X-Ca-
Signature":"[redacted]2uzWaQ=","Content-Type":"text/html; charset=utf-8","X-Ca-Signature-
Headers":"X-Ca-Timestamp,X-Ca-Key,X-Ca-Request-Mode,X-Ca-Stage,X-Ca-Supervisor-Token"}

Response:
404
Content-Type: application/oct-stream
Content-Length: 0
Connection: keep-alive
Date: Mon, 21 Dec 2020 02:52:44 GMT
X-Ca-Error-Message: API not found with `GET /test/`
X-Ca-Error-Code: I404NF
X-Ca-Request-Id: DA33D6E1-796F-4E16-B655-E8024602380E
Ali-Swift-Global-Savetime: 1608519164
Via: cache67.l2cn2655[173,404-1280,M], cache35.l2cn2655[174,0], vcache16.cn1070[246,404-1280,M], vcache6.cn1070[250,0]
X-Swift-Error: orig response 4XX error
X-Cache: MISS TCP_MISS dirn:-2:-2
```

2. Use X-Ca-Request-Id to query more API call information

The `X-Ca-Error-Code` and `X-Ca-Error-Message` headers help you identify the error cause. The `X-Ca-Request-Id` header helps you query request logs in Log Service or query API call results in the API Gateway console. You can also provide the request ID for the technical support personnel to help them check log information and resolve issues.

To query API call results in the API Gateway console, go to the Debug page. Specify `Region` and `X-Ca-Request-Id`, and click Search. The specific log information appears.



1. Method to get the error message

Once all API requests reach the API Gateway, the API Gateway returns the request result information. The user needs to look at the Header section that returns the result. In which, all the X-Ca words are gateway returns, and the important information is as follows:

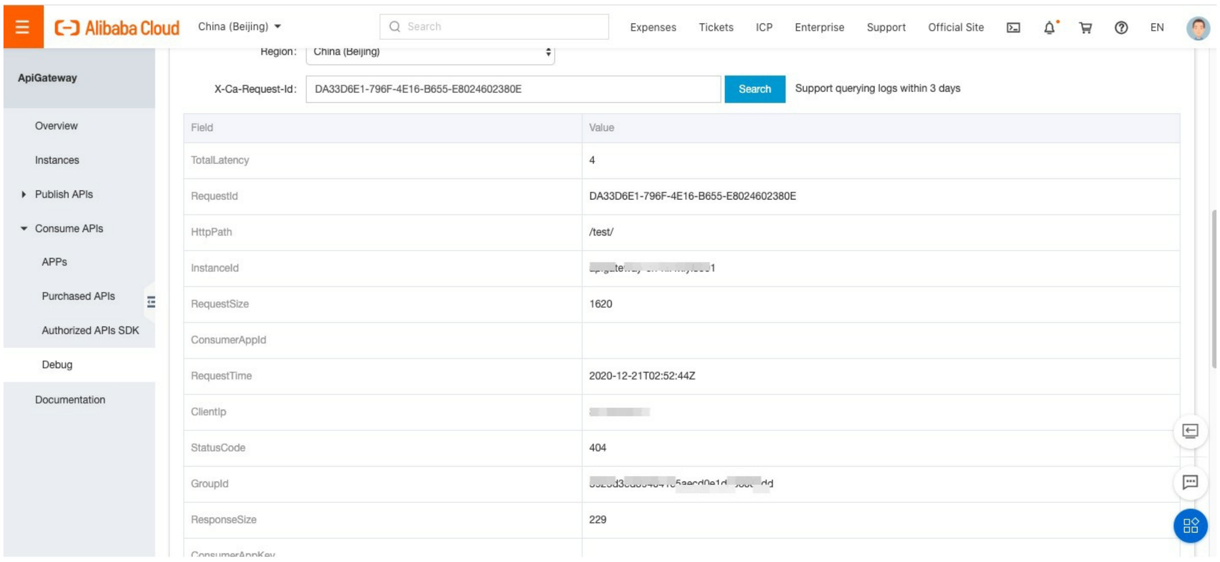
- X-Ca-Request-Id. Request unique ID. Once the request enters the API Gateway application, the API Gateway will generate the request ID and return it to the client through the response header. It is recommended that both the client and the back-end service record the request ID, which can be used for problem troubleshooting and tracking.
- X-Ca-Error-Message. An error message returned by the API Gateway, which returns the error message to the client via the response header when the request is in error.
- X-Ca-Error-Code. API Gateway system error code, error code provided by the API Gateway when the request is blocked by the gateway due to an error: (classic network instances do not have this header).

2. According to X-Ca-Request-Id query more call information

Region: China (Beijing)

X-Ca-Request-Id: DA33D6E1-796F-4E16-B655-E8024602380E Search Support querying logs within 3 days

3. Analyze the cause of the error



Region: China (Beijing)

X-Ca-Request-Id: DA33D6E1-796F-4E16-B655-E8024602380E Search Support querying logs within 3 days

Field	Value
TotalLatency	4
RequestId	DA33D6E1-796F-4E16-B655-E8024602380E
HttpPath	/test/
InstanceId	[redacted]
RequestSize	1620
ConsumerAppId	
RequestTime	2020-12-21T02:52:44Z
ClientIp	[redacted]
StatusCode	404
GroupId	[redacted]
ResponseSize	229
ConsumerAppKey	

You can also use Log Service to view API call logs. For more information about the fields in the logs, see Use Log Service to manage logs of API calls.

3. Analyze error causes

- - If the client receives a response in which the X-Ca-Error-Code header is empty, the HTTP status code of the response is generated by your backend service. API Gateway transparently transmits the error information from your backend service. In this case, check the backend service and network configurations between API Gateway and the backend service.
- - If the X-Ca-Error-Code header is not empty, the error code contained in the response is generated by API Gateway. The error code contains six characters in length.
- For more information about the error code, see error code tables.