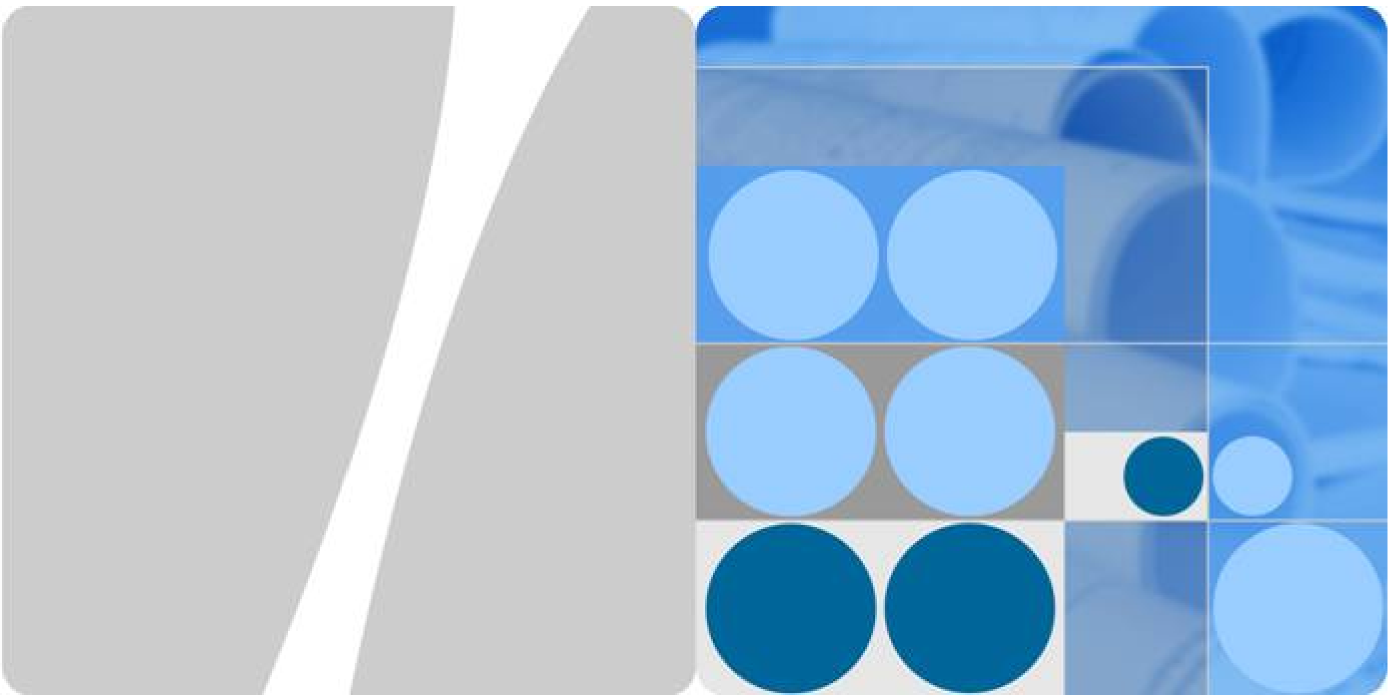


API Gateway

User Guide

Date **2024-10-09**

Contents

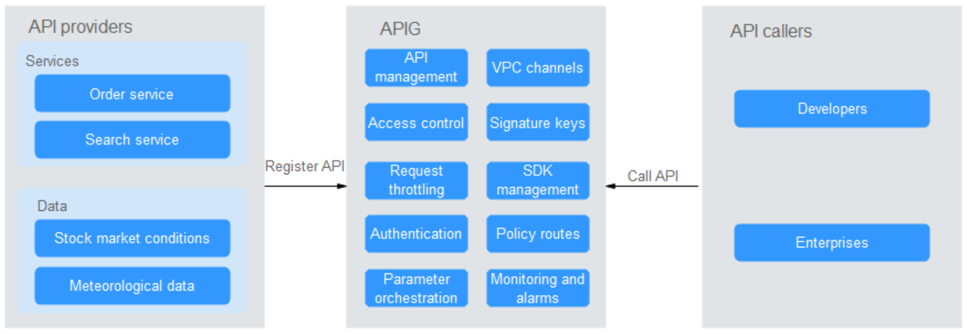
1 Service Overview

1.1 What Is APIG?

API Gateway (APIG) is your fully managed API hosting service. With APIG, you can build, manage, and deploy APIs at any scale to package your capabilities. With just a few clicks, you can integrate internal systems, monetize service capabilities, and selectively expose capabilities with minimal costs and risks.

- To monetize your capabilities (services and data), you can open them up by creating APIs in APIG. Then you can provide the APIs for API callers using offline channels.
- You can also obtain open APIs from APIG to reduce your development time and costs.

Figure 1-1 APIG architecture



Product Functions

- **API lifecycle management**

  The lifecycle of an API involves creating, publishing, removing, and deleting the API. API lifecycle management enables you to quickly and efficiently expose service capabilities.
- **Built-in debugging tool**

  With the built-in debugging tool, you can debug APIs using different HTTP headers and request bodies. This tool simplifies the API development process and reduces the API development and maintenance costs.

- **Version management**

  An API can be published in different environments. Publishing an API again in the same environment will override the API's previous version. APIG displays the publication history (including the version, description, date and time, and environment) of each API. You can roll back an API to any historical version to meet dark launch and version upgrade requirements.

- **Environment variables**

  Environment variables are manageable and specific to environments. Variables of an API will be replaced by the values of the variables in the environment where the API will be published. You can create variables in different environments to call different backend services using the same API.

- **Refined request throttling**
  - For different service demands and user levels, you can control the frequency at which an API can be called by a user, app (credential), or IP address, ensuring that backend services can run stably.
  - Configure different request throttling limits with API path, query, and header parameters.
  - The throttling can be accurate to the second, minute, hour, or day.
  - Set throttling limits for excluded applications (credentials) and tenants.

- **Monitoring and alarms**

  APIG provides visualized, real-time API monitoring, and displays multiple metrics, including number of requests, invocation latency, and number of errors. The metrics help you understand the API usage, allowing you to identify potential service risks.

- **Security**
  - Domain name access can be authenticated with TLS 1.1 and TLS 1.2. mTLS two-way authentication is supported.
  - Access control policies limit API access from specific IP addresses or accounts. You can blacklist or whitelist certain IP addresses and accounts to access your APIs.
  - Circuit breaker policies protect your backend services through degradation if they are abnormal.
  - Identity authentication can be based on AK/SK, function-based custom authorizers, and tokens. APIG verifies your backend services via certificates and is verified by your backend services through signature keys.

- **VPC channels (load balance channels)**

  Virtual Private Cloud (VPC) channels (load balance channels) can be created for accessing resources in VPCs and exposing backend services deployed in VPCs. VPC channels balance API requests to backend services.

- **Mock response**

  Mock backends simulate API responses for circuit breakers, service degradation, and redirection.

1.2 Product Advantages

Available Out-of-the-Box

You can quickly create APIs by configuring the required settings on the APIG console. APIG provides an inline debugging tool to simplify API development, and allows you to publish an API in multiple environments for easy testing and fast iteration.

Convenient API Lifecycle Management

APIG provides full-lifecycle API management, including design, development, test, publish, and O&M, to help you quickly build, manage, and deploy APIs at any scale.

Refined Request Throttling

APIG combines synchronous and asynchronous traffic control and multiple algorithms to throttle requests at the second level. You can flexibly define request throttling policies to ensure stability and continuity of API services.

Function Invocation

APIG seamlessly works with FunctionGraph, enabling you to selectively expose FunctionGraph functions in the form of APIs.

Visualized API Monitoring

APIG monitors the number of API calls, data latency, and number of errors, helping you identify potential service risks.

Comprehensive Security Protection

APIG provides multiple measures to secure API calling, such as Secure Sockets Layer (SSL) transfer, strict access control, IP address blacklist/whitelist, authentication, anti-replay, anti-attack, and multiple audit rules. In addition, APIG implements flexible and refined quota management and request throttling to help you flexibly and securely open your backend services.

Flexible Policy Routes

You can configure backends for an API to forward requests according to multiple policies. This facilitates dark launch and environment management.

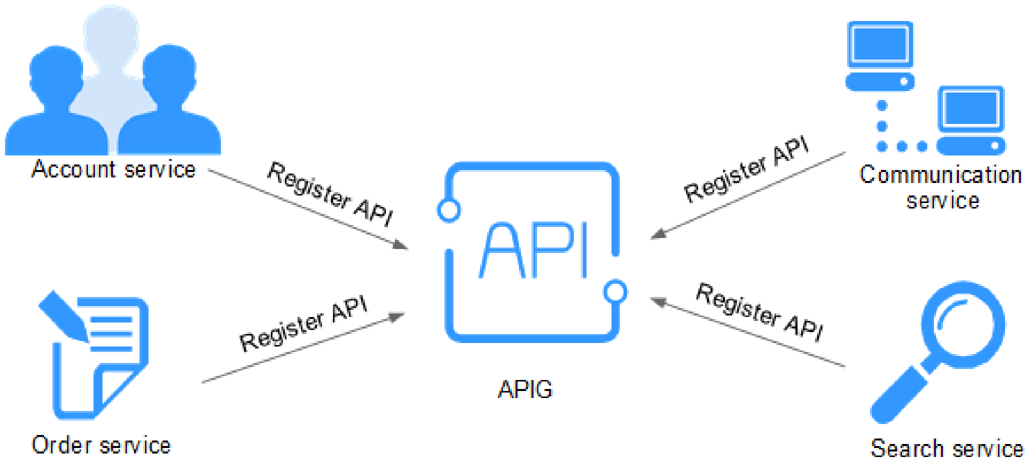
SDKs of Different Programming Languages

SDKs of different programming languages (such as Java, Go, Python, and C) are available for access from clients. Because the backends do not need to be modified, only one system is required to adapt to different service scenarios (such as mobile devices and IoT).

1.3 Application Scenarios

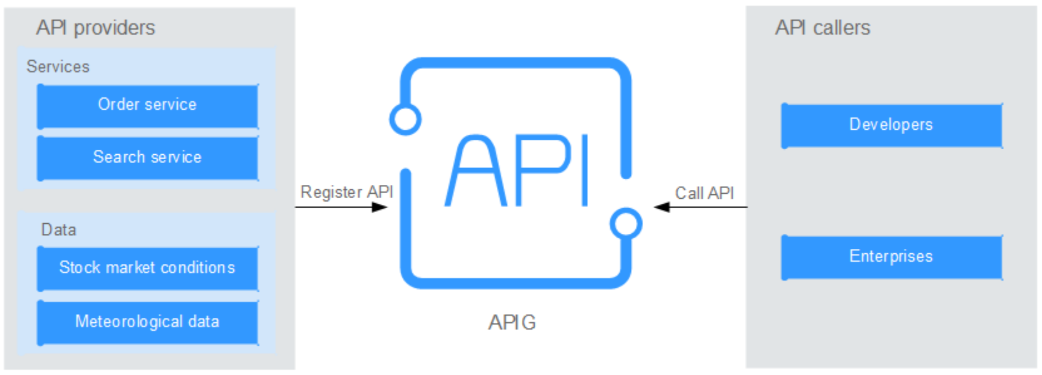
Internal System Decoupling

As enterprises develop rapidly with quick business changes, internal systems of enterprises need to keep pace with the development. However, it is difficult to ensure system universality and stability because internal systems are dependent on each other. APIG uses standard RESTful APIs to simplify the service architecture, decouples internal systems, and separates the frontend from backend. Existing capabilities can be reused to avoid repetitive development.



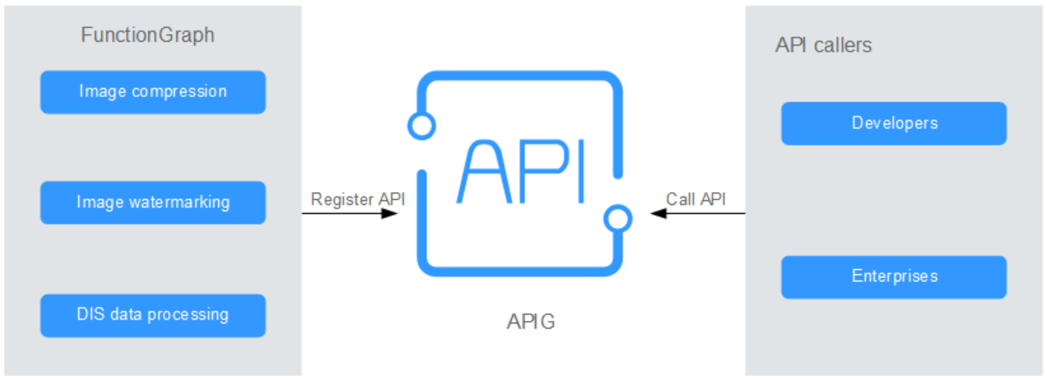
Enterprise Capabilities Opening

An enterprise cannot develop without partners' capabilities, such as a third-party payment platform and partner account login. APIG enables you to selectively expose capabilities to partners by using standard APIs and share services and data with partners to build a new ecosystem.



FunctionGraph Services Opening

APIG can also help you selectively expose serverless services (FunctionGraph services) to partners. FunctionGraph services are easier to develop, deploy, and maintain than traditional services. You can use FunctionGraph to quickly build backend service logic, and use APIG to expose service logic functions for linear concurrency expansion.



1.4 Specifications

Dedicated Gateway Specifications

The following table lists the specifications of dedicated gateways.

Table 1-1 Specifications of dedicated gateways

Edition	Maximum Number of Requests per Second
Basic	2000
Professional	4000
Enterprise	6000
Platinum	10,000

NOTE

- For dedicated gateways, you can adjust the maximum number of requests per second for each API.
- For details about how to modify the specifications of a dedicated gateway, see section "Modifying Specifications" in the *API Gateway User Guide*.

1.5 Notes and Constraints

Quota Limits

To change the default restrictions, increase the quota by referring to **Help Center > Others > FAQs > How Do I Apply for a Higher Quota?**

NOTICE

- It takes 5 to 10 seconds for a new or modified APIG resource to take effect.
- The maximum quota may be slightly exceeded in case of high concurrency, but resource usage will not be affected.

Table 1-2 Dedicated API gateway quotas

Item	Default Restriction	Modifiable
Gateways	5	√
API groups	1500	√
APIs	Number of APIs for each gateway edition: ● Basic: 250 ● Professional: 800 ● Enterprise: 2000 ● Platinum: 8000	√
APIs	1000 for each group	x
Backend policies	5	√
Apps (credentials)	50. The app quota includes the apps you have created.	√
Request throttling policies	● You can create a maximum of 300 request throttling policies for each gateway. ● The call limit for a single user cannot exceed that for the target API. ● The call limit for a single app (credential) cannot exceed that for a single user. ● The call limit for a single IP address cannot exceed that for the target API.	√
Environments	10	√
Signature keys	200	√
Access control policies	100	√
VPC channels (load balance channels)	200	√
Variables	You can create a maximum of 50 variables for an API group in each environment.	√
Independent domain names	A maximum of five independent domain names can be bound to an API group.	√

Item	Default Restriction	Modifiable
ECSs	A maximum of 10 ECSs can be added to a VPC channel.	√
Parameters	A maximum of 50 parameters can be created for an API.	√
API publication records	A maximum of 10 publication records of an API can be retained for each environment.	√
API access rate	Up to 6000 times per second	√
Excluded applications (Credentials)	A maximum of 30 excluded apps can be added to a request throttling policy.	√
Excluded tenants	A maximum of 30 excluded tenants can be added to a request throttling policy.	√
Access to a subdomain name (debugging domain name)	A subdomain name can be accessed up to 1000 times a day.	x
Maximum size of an API request package	12 MB	√
TLS protocol	TLS 1.1 and TLS 1.2 are supported. TLS 1.2 is recommended.	√
Custom authorizers	50	x
Plug-ins	500	√
HTTP protocol	When the HTTP protocol is used, the maximum size of URL+Header is 32 KB.	x

1.6 Permissions Management

If you need to assign different permissions to personnel in your enterprise to access your APIG resources, Identity and Access Management (IAM) is a good choice for fine-grained permissions management. IAM provides identity authentication, permissions management, and access control, helping you securely access your resources.

With IAM, you can use your account to create IAM users for your employees, and assign permissions to the employees to control their access to specific resources.

If your account does not require individual IAM users for permissions management, skip this chapter.

APIG Permissions

By default, new IAM users do not have any permissions assigned. You need to add a user to one or more groups, and attach policies or roles to these groups. The user then inherits permissions from the groups to which the user belongs, and can perform specified operations on cloud services based on the permissions.

APIG is a project-level service deployed and accessed in specific physical regions. To assign APIG permissions to a user group, you need to specify region-specific projects for which the permissions will take effect. If you select **All projects**, the permissions will be granted for both the global service project and all region-specific projects. When accessing APIG, the users need to switch to a region where they have been authorized to use this service.

You can grant permissions by using roles and policies.

- Roles: A type of coarse-grained authorization mechanism that defines permissions related to user responsibilities. This mechanism provides only a limited number of service-level roles for authorization. When using roles to grant permissions, you need to also assign other dependent roles for permissions to take effect. However, roles are not an ideal choice for fine-grained authorization and secure access control.
- Policies: A fine-grained authorization strategy that defines permissions required to perform operations on specific cloud resources under certain conditions. This mechanism allows for more flexible policy-based authorization and meets requirements for secure access control. For example, you can grant APIG users only the permissions for performing specific operations. Most policies define permissions based on APIs. For the API actions supported by APIG, see section "Permissions Policies and Supported Actions" in the *API Reference*

Table 1-3 lists all the system-defined roles and policies supported by APIG.

Table 1-3 System-defined roles and policies supported by APIG

Role/ Policy Name	Description	Type	Dependency
APIG Administrator	Administrator permissions for APIG. Users with this permission can use all functions.	System-defined role	If a user needs to create, delete, or change resources of other services, the user must also be granted administrator permissions of the corresponding services in the same project.

Role/ Policy Name	Description	Type	Dependency
APIG FullAccess	Full permissions for APIG. Users granted these permissions can use all functions of gateways.	System-defined policy	None
APIG ReadOnly Access	Read-only permissions for APIG. Users granted these permissions can only view gateways.	System-defined policy	None

You can view the content of the preceding roles and policies on the IAM console. For example, the content of the **APIG FullAccess** policy is as follows:

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Action": [
                "apig:*:*",
                "vpc:*:get*",
                "vpc:*:list*",
                "vpc:ports:create",
                "vpc:ports:update",
                "vpc:ports:delete",
                "vpc:publicIps:update",
                "FunctionGraph:function:listVersion",
                "FunctionGraph:function:list",
                "FunctionGraph:function:getConfig",
                "ecs:servers:list",
                "lts:groups:list",
                "lts:logs:list",
                "lts:topics:list"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Related Documents

- Section "Service Overview" in the *Identity and Access Management User Guide*
- Section "Creating a User and Granting Permissions" in the *API Gateway User Guide*

1.7 Basic Concepts

API

A set of predefined functions that encapsulates application capabilities. You can create APIs and make them accessible to users.

When creating an API, you need to configure the basic information and the frontend and backend request paths, parameters, and protocols.

API Group

A collection of APIs used for the same service. API groups facilitate API management.

Environment

A stage in the lifecycle of an API. An environment, such as API testing or development environment, specifies the usage scope of APIs, facilitating API lifecycle management. The same API can be published in different environments.

To call an API in different environments, you need to add the **x-stage** header parameter to the request sent to call the API. The value of this parameter is an environment name.

Environment Variable

A variable that is manageable and specific to an environment. You can create variables in different environments to call different backend services using the same API.

Request Throttling

Controls the number of times APIs can be called by a user, app (credential), or IP address during a specific period to protect backend services.

Request throttling can be accurate to the minute and second.

Access Control

Access control policies are one of the security measures provided by APIG. They allow or deny API access from specific IP addresses or accounts.

App (Credential)

An entity that requests for APIs. An app can be authorized to access multiple APIs, and multiple apps can be authorized to access the same API.

Signature Key

Consists of a key and secret, which are used by backend services to verify the identity of API Gateway and ensure secure access.

When an API bound with a signature key is called, API Gateway adds signature information to the API requests. The backend service of the API signs the requests in the same way, and verifies the identity of API Gateway by checking whether the signature is consistent with that in the **Authorization** header sent by API Gateway.

VPC Channel (Load Balance Channel)

A method for accessing VPC resources from API Gateway, allowing you to selectively expose backend services deployed in VPCs to third-party users.

Custom Authentication

A mechanism defined with custom rules for API Gateway to verify the validity and integrity of requests initiated by API callers. The mechanism is also used for backend services to verify the requests forwarded by API Gateway.

The following two types of custom authentication are provided:

- Frontend custom authentication: A custom authorizer is configured with a function to authenticate requests for an API.
- Backend custom authentication: A custom authorizer can be configured to authenticate requests for different backend services, eliminating the need to customize APIs for different authentication systems and simplifying API development. You only need to create a function-based custom authorizer in API Gateway to connect to the backend authentication system.

Simple Authentication

Simple authentication facilitates quick response for API requests by adding the **X-Apig-AppCode** parameter (whose value is an AppCode) to the HTTP request header. API Gateway verifies only the AppCode and does not verify the request signature.

Gateway Response

Gateway responses are returned if API Gateway fails to process API requests. API Gateway provides default responses for multiple scenarios and allows you to customize response status codes and content. You can add a gateway response in JSON format on the **API Groups** page.

1.8 Billing

APIG helps you build, manage, and deploy APIs at any scale.

To learn about the pricing of APIG and calculate the prices for using this service, go to the **Product Pricing Details** page.

Billing

Gateways are billed based on the **gateway edition** and **bandwidth**.

Billing for the Gateway Edition

Gateways are available in four editions: basic, professional, enterprise, and platinum. You need to pay the corresponding prices when purchasing these editions.

- Pay-per-use (hourly): You can start and stop gateways as needed. You will be billed based on the duration for which you use the gateways. Billing starts when a gateway is purchased and ends when the gateway is stopped due to arrears or is deleted. The minimum time unit is one second.

Billing for Bandwidth

If your API backend service is deployed on the public network, you will be charged for the bandwidth for forwarding API requests to the public network. The prices are calculated based on the **bandwidth** and the **duration** for which you use the gateway.

NOTE

- If your backend service is deployed in the same VPC as your gateway, the backend service can be accessed using a private IP address, and you do not need to purchase bandwidth for the gateway.
- If your gateway contains APIs that will be called from public networks, you need to purchase an EIP and bind it to the gateway.
- If the APIs in your gateway will be called within a VPC, you do not need to purchase or bind an EIP to the gateway.

2 Getting Started

2.1 Introduction

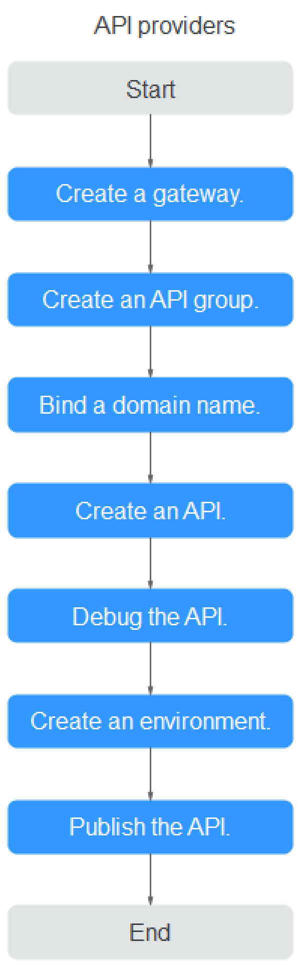
API Gateway (APIG) is a fully managed service that enables you to securely build, manage, and deploy APIs at any scale with high performance and availability. With APIG, you can easily integrate your internal service systems and selectively expose your service capabilities.

To learn about the process of exposing and calling an API, see **Opening APIs** and **Calling APIs**. **Simple authentication** with an app is used for illustration.

2.2 Opening APIs

2.2.1 Process Flow

The following figure shows the process of exposing an API.



1. Creating a Gateway
   Buy a gateway.
2. **Creating an API Group**
   An API group facilitates management of APIs used for the same service. Create an API group and then create APIs.
3. **Binding a Domain Name**
   Before making the API available for users to access, bind an independent domain name (custom domain name) to the group to which the API belongs. Then API callers can use these domain names to call the API.
4. **Creating an API**
   When creating an API, configure the frontend and backend request paths, parameters, and protocols.
5. **Debugging an API**
   Debug the API to check whether it works normally.
6. **(Optional) Creating an Environment**
   An API can be called in different scenarios, such as the production environment (RELEASE) or other custom environments. RELEASE is the default environment defined in APIG.
7. **Publishing an API**
   Publish the API so that it can be called.

2.2.2 Creating an API Group

Step 1 Log in to the APIG console.

Step 2 In the upper part of the navigation pane, select **the gateway**.

Step 3 In the navigation pane, choose **API Management** > **API Groups**.

Step 4 Choose **Create API Group** > **Create Directly**.

Table 2-1 API group information

Parameter	Description
Name	API group name. It is recommended that you enter a name based on naming rules to facilitate search.
Description	Description of the API group.

Step 5 Click **OK**. The system automatically allocates a debugging domain name to the API group. APIs in the group can be debugged using the domain name.

----End

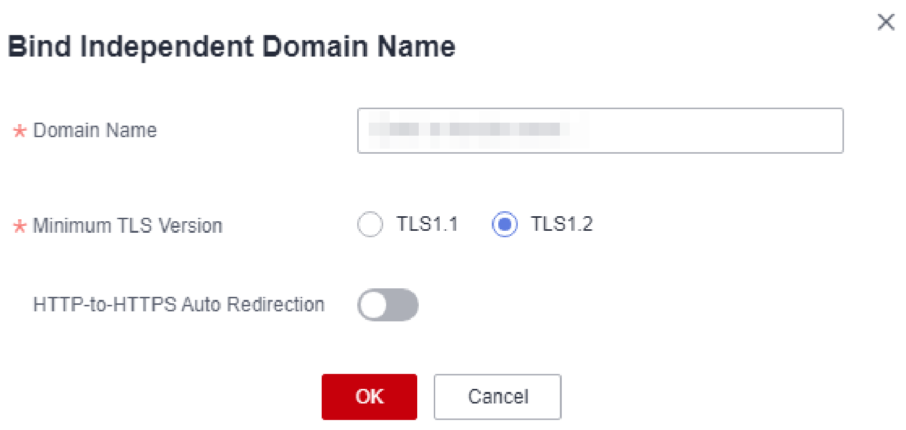
2.2.3 Binding a Domain Name

Step 1 On the **API Groups** page, click the group created in **Creating an API Group** to go to the group details page.

Step 2 Click the **Group Information** tab.

Step 3 Click **Bind Independent Domain Name** in the **Independent Domain Names** area.

Figure 2-1 Binding an independent domain name


Bind Independent Domain Name

* Domain Name: [redacted]

* Minimum TLS Version: TLS1.1 [✓ TLS1.2]

HTTP-to-HTTPS Auto Redirection: off

NOTE

The independent domain name must be registered and resolved. For details, see "Prerequisites" in **Binding a Domain Name**.

----End

2.2.4 Creating an API

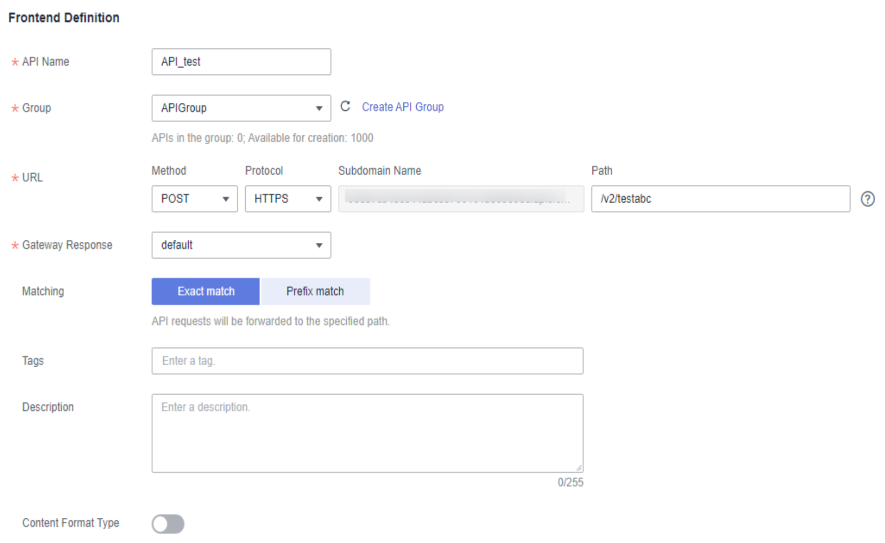
Procedure:

1. **Configuring Frontend Settings**
2. **Configuring Backend Settings**

Configuring Frontend Settings

Step 1 In the navigation pane, choose **API Management** > **APIs**.

Step 2 Click **Create API** > **Create API** and configure the frontend.



Frontend Definition

* API Name	API_test
* Group	APIGroup ▾ (Create API Group) APIs in the group: 0; Available for creation: 1000
* URL	Method: POST; Protocol: HTTPS; Subdomain Name: [redacted]; Path: /v2/testabc
* Gateway Response	default
Matching	[✓ Exact match] Prefix match — API requests will be forwarded to the specified path.
Tags	Enter a tag.
Description	Enter a description. 0/255
Content Format Type	

Table 2-2 Frontend definition

Parameter	Description
Name	API name. It is recommended that you enter a name based on naming rules to facilitate search.
Group	By default, the group created in **Creating an API Group** is selected.
URL	**Method**: Request method of the API. Set this parameter to **POST**. **Protocol**: Set this parameter to **HTTPS**. Subdomain name: The subdomain automatically allocated to the API group created in **Creating an API Group**. **Path**: Path for requesting the API.

Parameter	Description
Gateway Response	Select a response to be displayed if API Gateway fails to process an API request. The default gateway response is **default**.
Matching	By default, **Exact match** is selected.
Tags	Classification attribute used to quickly identify the API from other APIs.
Description	Description of the API.

Step 3 Configure security settings based on the following table.

Table 2-3 API request definition

Parameter	Description
Authentication Mode	API authentication mode. Set this parameter to **App**.
Simple Authentication	If you enable this option, API Gateway verifies only the AppCode and the request signature does not need to be verified. For this example, enable simple authentication.

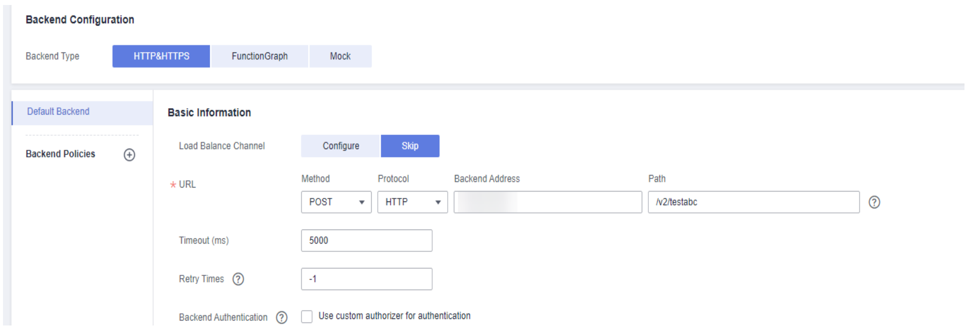
Step 4 Click **Next**.

----End

Configuring Backend Settings

Step 1 On the **Backend Configuration** page, set the backend service information.

Step 2 Select a backend service type. For this example, select **HTTP&HTTPS**.


Backend Configuration

Backend Type: HTTP&HTTPS | FunctionGraph | Mock

Default Backend

Backend Policies

Basic Information

Load Balance Channel: Configure | Skip

* URL

Method	Protocol	Backend Address	Path
POST	HTTP	[redacted]	/v2/testabc

Timeout (ms): 5000

Retry Times: -1

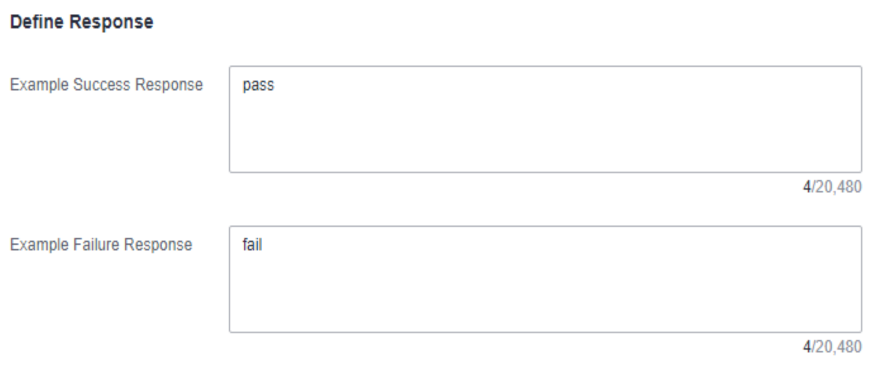
Backend Authentication: Use custom authorizer for authentication

Table 2-4 HTTP&HTTPS backend service definition

Parameter	Description
Load Balance Channel	Determine whether the backend service will be accessed using a load balance channel. For this example, select **Skip**.

Parameter	Description
URL	**Method**: Request method of the API. Set this parameter to **POST**. **Protocol**: Set this parameter to **HTTP**. **Backend Address**: Address of the backend service. **Path**: Path of the backend service.
Timeout	Backend service request timeout. Default value: 5000 ms.

Step 3 On the **Define Response** page, set the responses.


Define Response

Example Success Response: pass

4/20,480

Example Failure Response: fail

4/20,480

Table 2-5 Defining responses

Parameter	Description
Example Success Response	An example of a response returned when the API is called successfully.
Example Failure Response	An example of a response returned when the API fails to be called.

Step 4 Click **Finish**.

----End

2.2.5 Debugging an API

Step 1 On the **APIs** tab page, select an API from **Creating an API** and click **Debug**.

Step 2 Configure the URL.

Step 3 Click **Debug**. The API request and response information are displayed at the bottom of the page.

If the API is called successfully, the status code **200** is displayed. Otherwise, rectify the fault by referring to **Error Codes**.

----End

2.2.6 (Optional) Creating an Environment

Step 1 In the navigation pane, choose **API Management** > **API Policies**. Then click the **Environments** tab.

Step 2 Click **Create Environment** and set the environment information.

Table 2-6 Environment information

Parameter	Description
Name	Environment name. It is recommended that you enter a name based on naming rules to facilitate search.
Description	Description of the environment.

Step 3 Click **OK**.

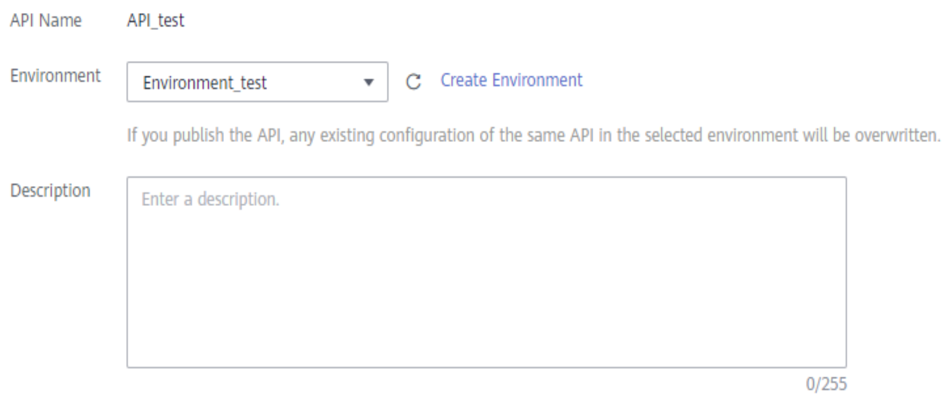
----End

2.2.7 Publishing an API

Step 1 In the navigation pane, choose **API Management** > **APIs**.

Step 2 Locate the API created in **Creating an API**, and click **Publish**.

Step 3 Select the environment where the API will be published.



API Name API_test

Environment Environment_test

If you publish the API, any existing configuration of the same API in the selected environment will be overwritten.

Description Enter a description.

0/255

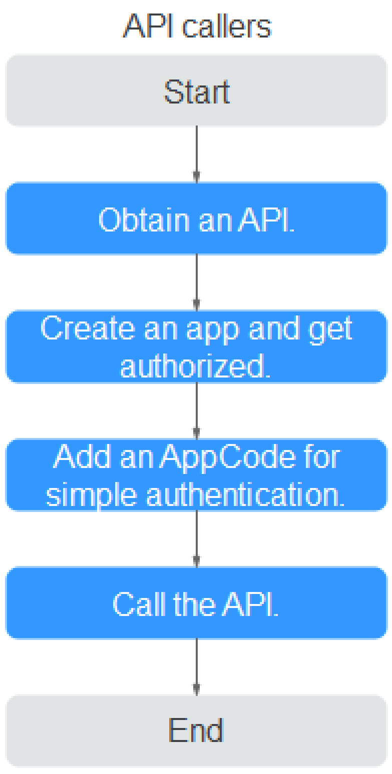
Step 4 Click **OK**.

----End

2.3 Calling APIs

2.3.1 Process Flow

The following figure shows the process of calling an API.


API callers
Start
Obtain an API.
Create an app and get authorized.
Add an AppCode for simple authentication.
Call the API.
End

1. Obtaining an API

   Obtain an API and its documentation from an API provider.
2. **Creating a Credential and Getting Authorized**

   APIs that use app authentication can only be called using credentials bound to them.
3. **Adding an AppCode for Simple Authentication**

   API Gateway only verifies the AppCode during simple authentication.
4. **Calling an API**

   Use an API test tool to call the API with the simple authentication AppCode.

2.3.2 Creating a Credential and Getting Authorized

Creating a Credential

Step 1 In the navigation pane, choose **API Management** > **Credentials**.

Step 2 Click **Create Credential** and set credential information.

Table 2-7 Credential information

Parameter	Description
Name	Credential name. It is recommended that you enter a name based on naming rules to facilitate search.
Description	Description about the credential.

Step 3 Click **OK**.

----End

Binding to an API

Step 1 In the **Operation** column of the **created credential**, click **Bind to APIs**. **Note that only APIs that use app authentication can be bound.**

Step 2 Select the environment, API group, and API created in **Opening APIs**, and click **OK**.

----End

2.3.3 Adding an AppCode for Simple Authentication

Step 1 In the credential list, click the credential created in **Creating a Credential** to go to the credential details page.

Step 2 Click **Add AppCode** in the **AppCodes** area.

Step 3 Select **Automatically generated**.

Step 4 Click **OK**.

----End

2.3.4 Calling an API

Use the API test tool to configure the API request and authentication. For details about how to call an API, see section "Calling Published APIs" > **Calling APIs**.

Step 1 Obtain the API request information and construct the API URL.

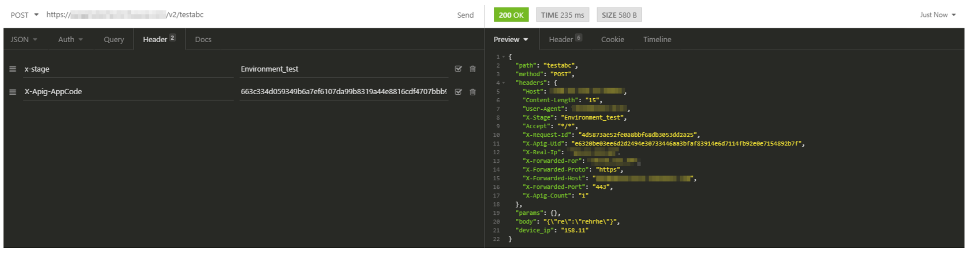
For illustration purposes, an API and its documentation are obtained through offline channels. You can also obtain the authentication mode, request method, request path, and other information about the API.

Step 2 Add the header parameter **X-Apig-AppCode** and set the parameter value to the **generated AppCode**.

Step 3 Add the header parameter **x-stage** and set the parameter value to the **running environment**. Skip this step if the API has been published in the RELEASE environment.

Step 4 Click **Send** to send a request.

If the API is called successfully, **200 OK** is displayed. Otherwise, rectify the fault by referring to **Error Codes**.



POST https://[redacted]/v2/testabc

x-stage	Environment_test
X-Apig-AppCode	663c334d059349b6a7ef6107da99b8319a44e8816cdf4707bbb!

```
{
  "path": "testabc",
  "method": "POST",
  "headers": {
    "Host": [redacted],
    "Content-Length": "15",
    "User-Agent": [redacted],
    "X-Stage": "Environment_test",
    "Accept": "*/*",
    "X-Request-Id": "4d5873ae52fe0a8bbf68db3053dd2a25",
    "X-Apig-Uid": "e6320be03ee6d2d2494e30733446aa3bfaf83914e6d7114fb92e0e7154892b7f",
    "X-Real-Ip": [redacted],
    "X-Forwarded-For": [redacted]
    "X-Forwarded-Proto": "https",
    "X-Forwarded-Host": "[redacted]",
    "X-Forwarded-Port": "443",
    "X-Apig-Count": "1"
  },
  "params": {},
  "body": "{\"re\":\"rehrhe\"}",
  "device_ip": "158.11"
}
```

----End

3 Comparing Versions

The new API Gateway (APIG) console is available since April 2, 2023. The following table lists the differences between the old and new consoles.

Table 3-1 Comparing versions

Difference	Old	New
Two-factor authentication	Not supported	Supported
Configuration of retries for HTTP&HTTPS backend services	Not supported	Supported
API import	For registering APIs	For registering APIs or creating API groups
API debugging with a custom body	Not supported	Supported
API details page	Average	Highly integrated
API topology	Supported	Not supported
Visual display of API policies	Not supported	Supported
Creating policy by script	Not supported	Supported
Plug-in type	CORS, HTTP response header management, request throttling	CORS, HTTP response header management, request throttling 2.0, Kafka log push, circuit breaker. These plug-ins are managed together with traditional policies.
SSL certificate management	Not supported	Supported
Creating server groups for load balance channels	Not supported (VPC channels)	Supported

Difference	Old	New
Health check switch for load balance channels	Not supported (VPC channels)	Supported
Display of standby nodes and statuses for load balance channels	Not supported (VPC channels)	Supported
Apps	Supported	Now called "credentials"
Credential quota policies	Not supported	Supported
Access control policies	Not supported	Supported
API monitoring	Provided on the **Dashboard** page	Now called "API Monitoring"
Subdomain name	Supported	Now called "debugging domain name"
Variable management	Variables	Now called **Environment Variables**
Gateway selection in left navigation pane	Not supported	Supported
Bandwidth monitoring	Not supported	Supported
Importing CCE workloads	Not supported	Supported
Gateway tags	Not supported	Supported
Microservice load balance channels	Not supported	Supported
Account ID-based access control	Not supported	Supported
Cloning APIs	Not supported	Supported
custom_log	Not supported	Supported
Third-party authentication policy	Not supported	Supported
gzip compression	Not supported	Supported
gRPC protocol	Not supported	Supported
Server-sent events (SSE) transmission	Not supported	Supported
API policy cloning	Not supported	Supported
OpenAPI 3.0	Not supported	Supported

Difference	Old	New
Client request customization	Not supported	Supported
HTTP-to-HTTPS auto redirection	Not supported	Supported
CA certificate	Not supported	Supported

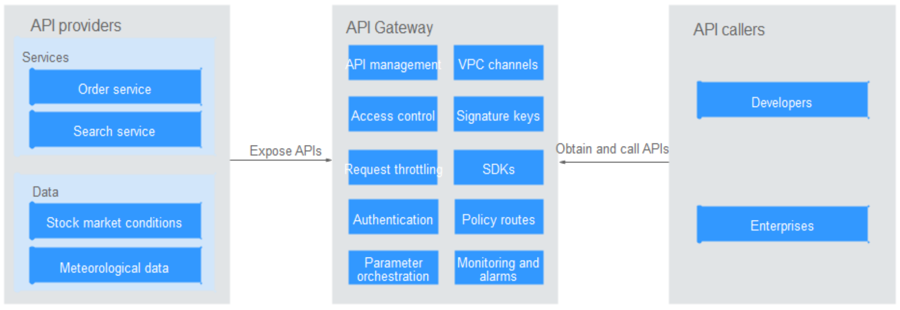
4 Overview

APIG is a fully managed service that enables you to securely build, manage, and deploy APIs at any scale with high performance and availability. With APIG, you can easily integrate your internal service systems and selectively expose and monetize your service capabilities.

General Procedure

The following figure shows the procedure for using APIG to host APIs.

Figure 4-1 APIG

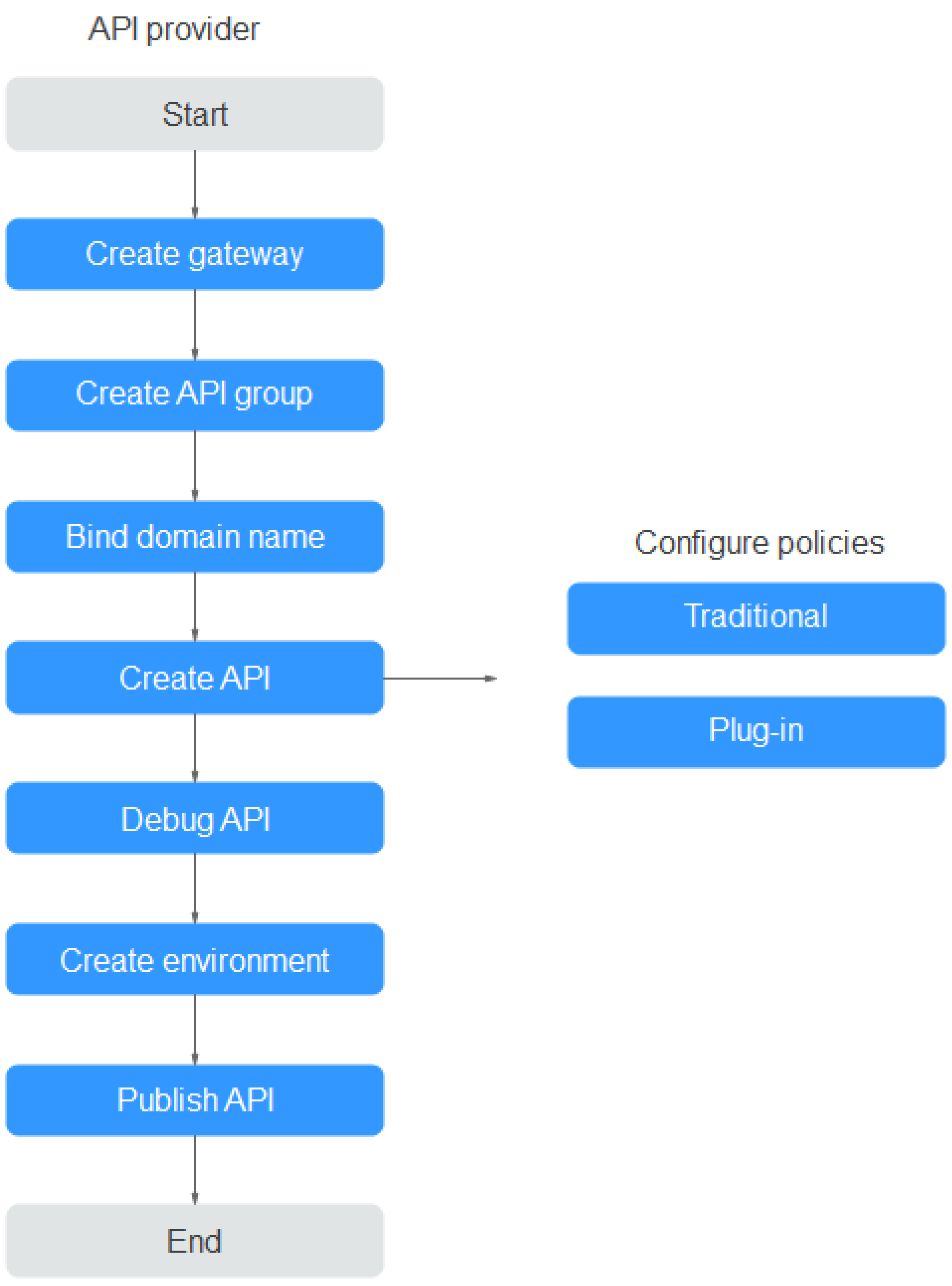


You can **expose your API services** or **obtain and call APIs of others** through APIG.

Exposing APIs

Enterprises or developers selectively expose and monetize their services and data through APIG.

Figure 4-2 API exposing process



1. **Create a gateway.**

   A gateway is an independent resource space where all operations are performed. Resources of different gateways are isolated from each other.

2. **Create an API group**.

   Each API belongs to an API group. Create an API group before creating an API.

3. **Bind a domain name.**

   Before exposing an API, bind an independent domain name to the target group so that API callers can access the API.

You can debug the API using the debugging domain name allocated to the group to which the API belongs. The domain name can be accessed a maximum of 1000 times every day.

4. **Create an API.**

   Encapsulate existing backend services into standard RESTful APIs and expose them to external systems.

   After creating an API, configure the following settings to control API access:

   – Traditional policies

     - **Request throttling**

       Request throttling controls the number of times an API can be called within a time period to protect backend services.

     - **Access control**

       Set a blacklist or whitelist to deny or allow API access from specific IP addresses or accounts.

     - **Signature keys**

       Signature keys are used by backend services to verify the identity of APIG.

   – Plug-in policies

     - **CORS**

       This policy provides the capabilities of specifying preflight request headers and response headers and automatically creating preflight request APIs for cross-origin API access.

     - **HTTP Response Header Management**

       You can customize HTTP response headers that will be contained in an API response.

     - **Request Throttling 2.0**

       This policy enables you to limit the number of times an API can be called within a specific time period. Parameter-based, basic, and excluded throttling is supported.

     - **Kafka Log Push**

       This policy pushes API calling logs to Kafka so that users can easily obtain them.

     - **Circuit Breaker**

       This policy protects your backend service when a performance issue occurs.

     - **Third-Party Authorizer**

       You can configure your own service to authenticate API requests.

5. **Debug the API.**

   Verify whether the API is working normally.

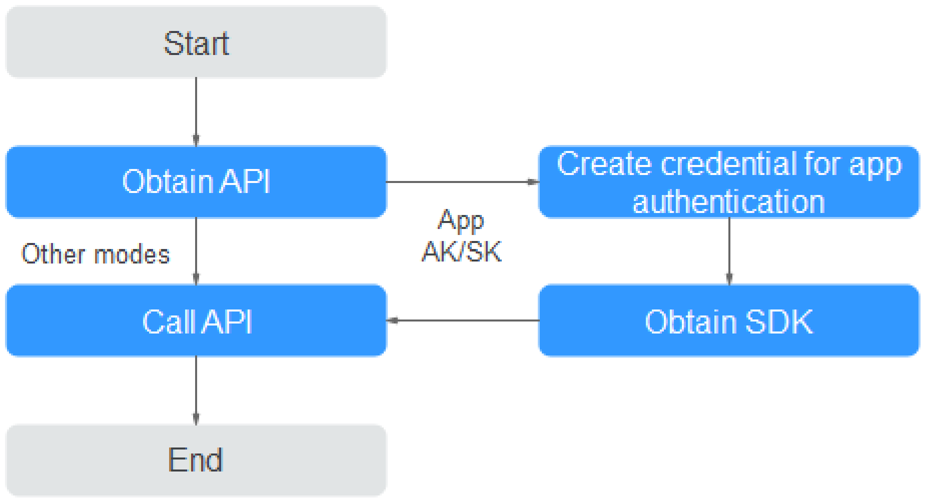
6. **Publish the API.**

   The API can be called only after it has been published in an environment.

Calling APIs

Enterprises and developers obtain and call APIs of other providers, thereby reducing development time and costs.

Figure 4-3 API calling process



1. **Obtain an API.**

   Obtain the API request information, including the domain name, protocol, method, path, and authentication mode.

2. **Create a credential.**

   For an API that uses app authentication, create a credential to generate a credential ID and key/secret pair. Bind the credential to the API so that you can call the API through app authentication.

3. **Obtain an SDK**.

   Use the SDK to generate a signature for the AK/SK and call the API.

4. **Call the API.**

   Call the API using its access address and perform authentication based on its authentication mode.

5 API Management

5.1 Creating an API Group

An API group contains APIs used for the same service. You can manage APIs by group, and must create a group before creating an API.

You can create an API group using the following methods:

- **Creating an API Group Directly**

  You can create APIs for the group as required.
- **Importing an API Design File**

  Import an API file to create a group.
- Importing a CCE Workload

  By importing Cloud Container Engine (CCE) workloads, you can open up your CCE service capabilities. For details, see **Importing a CCE Workload**.

NOTE

- To make your APIs available for users to access, bind independent domain names to the group to which the APIs belong.
- Each API can belong to only one group.
- The system automatically allocates a subdomain name to each API group for internal testing. The subdomain name can be accessed 1000 times a day. **You can also disable the Debugging Domain Name switch. When disabled, the debugging domain name is hidden and APIs cannot be called through it.**
- API group **DEFAULT** is automatically generated for each gateway. APIs in this group can be called using the IP address of the Virtual Private Cloud (VPC) where the gateway is deployed.

Prerequisites

You have **created a gateway**.

Creating an API Group Directly

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 Choose **API Management** > **API Groups**.

Step 4 Choose **Create API group** > **Create Directly**, and enter group information.

Table 5-1 Group information

Parameter	Description
Name	API group name.
Description	Description of the API group.

Step 5 Click **OK**.

----End

Importing an API Design File

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 Choose **API Management** > **API Groups**.

Step 4 Choose **Create API Group** > **Import API Design File**.

Step 5 Select an API file and click **Open**.

Step 6 Set the import parameters.

Table 5-2 Parameters for importing APIs

Parameter	Description
Import	Options: ● **New group**: Import APIs to a new API group. If you select this option, the system automatically creates an API group and imports the APIs into this group. ● **Existing group**: Import APIs to an existing API group. If you select this option, the system adds the APIs to the selected API group while retaining the existing APIs in the API group.
API group	Select an API group if you set **Import** to **Existing group**.
Basic Definition Overwrite	Determine whether to overwrite an existing API if the name of the API is the same as that of an imported API. This parameter is available only if you set **Import** to **Existing group**.
Extended Definition Overwrite	If this option is selected, the extended definition items (access control and request throttling policies) of an imported API will overwrite the existing policies with the same name.

Step 7 (Optional) To configure the APIs, click **Configure Global Settings**.

1. Change the authentication mode. For details, see **5.2**.
2. Modify the backend request configuration. For details, see **Step 1**.
3. Click **Next**. You can view the configuration details in form, JSON, or YAML format.
4. Confirm the settings and click **Submit**.

Step 8 Click **Import Now**, and determine whether to publish the APIs.

- **Now**: Publish the APIs in a specified environment now.
- **Later**: **Publish the APIs** later.

Step 9 Click **OK**. The **APIs** tab is displayed, showing the imported APIs.

----End

Follow-Up Operations

After an API group is created, **bind independent domain names** to it so that API callers can use them to call open APIs in the group.

5.2 Importing a CCE Workload

By importing Cloud Container Engine (CCE) workloads, you can open your CCE service capabilities through APIs.

NOTE

If your gateway does not support CCE workload import, contact customer service.

Precautions

- Only CCE Turbo clusters and CCE clusters using the VPC network model are supported.
- The CCE cluster and your gateway must be in the same VPC or otherwise connected.
- If you select a CCE cluster that uses a VPC network model, add the container CIDR block of the cluster to **Routes** on the gateway details page.
- After the import, APIs will be generated, together with a microservice load balance channel that monitors and updates address changes of all pods in the workload.

Prerequisites

You have created CCE workloads. For details, see section "Creating a Deployment" in the *CCE User Guide*.

Procedure

Step 1 Go to the APIG console.

Step 2 Select a gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **API Management** > **API Groups**.

Step 4 Choose **Create API Group** > **Import CCE Workload**. Set the parameters according to the following table.

Table 5-3 Parameter description

Parameter	Description
Group	Group to which the CCE workload belongs. You can create a group or select an existing group.
Cluster	Select a cluster. Click **View CCE Console** to view the available clusters.
Namespace	Namespace to which the workload will belong. A namespace is an abstract collection of resources and objects.
Workload Type	● **Deployment**: Deployments do not store any data or status while they are running. ● **StatefulSet**: StatefulSets store data and statuses while they are running. ● **DaemonSet**: DaemonSets ensure that only one pod runs on all or some nodes. When a node is added to a cluster, a new pod is also added for the node. When a node is removed from a cluster, the pod is also reclaimed. If a DaemonSet is deleted, all pods created by it will be deleted. For details about workload types, see "Overview" in the *CCE User Guide*.
Service Label Key	Pod label of a workload. The service label name is the pod label key and the service label value is the pod label value. For details about pod labels, see section "Labels and Annotations" in the *CCE User Guide*.
Service Label Value	
Tag	Pod label of a workload. If a workload cannot be identified by certain service label name and value, select another pod label to specify the workload.
Protocol	**HTTP** and **HTTPS** are supported. **HTTPS** is recommended for transmitting important or sensitive data.
Request Path	You can use a plus sign (+) for prefix matching. For example, **/a/{b+}**.
Port	Listening port of the CCE workload.
Authentication Mode	App and IAM authentication is supported. You can also choose not to authenticate requests. ● **App**: Requests will be authenticated by APIG. This authentication mode is recommended. ● **IAM**: Requests will be authenticated by IAM. ● **None**: No authentication will be required.

Parameter	Description
CORS	Determine whether to enable cross-origin resource sharing (CORS). CORS allows browsers to send XMLHttpRequest to servers in other domains, overcoming the limitation that Asynchronous JavaScript and XML (AJAX) can be used only within the same domain. There are two types of CORS requests: ● Simple requests: requests that have the **Origin** field in the header. ● Not-so-simple requests: HTTP requests sent before the actual request. If CORS (not-so-simple request) is enabled for an API, another API that uses the OPTIONS method must be created. For details, see **Enabling CORS**.
Timeout (ms)	Backend request timeout. If a backend timeout error occurs during API debugging, increase the timeout to locate the reason. **NOTE** Modify the maximum timeout by referring to **Configuring Parameters**. The value range is 1 ms to 600,000 ms.

Step 5 Click **OK**.

----End

5.3 Binding a Domain Name

Before you expose an API, bind an independent domain name to the API group so that APIs in the group can be accessed with the domain name. A maximum of five independent domain names can be added, and the number of access times is not limited.

Independent domain names are classified into private and public domain names.

- Private domain name: Service systems deployed on the cloud service platform can use private domain names to access APIs.
- Public domain name: Service systems deployed outside the cloud service platform can use public domain names to access APIs.

For internal testing, use the debugging domain name (subdomain name) to access APIs in an API group (maximum 1,000 times a day). The debugging domain name cannot be modified.

NOTE

- Groups under the same gateway cannot be bound with a same independent domain name.
- By default, the debugging domain name of an API group can only be resolved to a server in the same VPC as the gateway. If you want to resolve the domain name to a public network, bind an EIP to the gateway.
- If the independent domain name you select is a wildcard domain name (for example, ***.aaa.com**), you can use any of its subdomain names (for example, **default.aaa.com** and **1.aaa.com**) to access all APIs in the group to which the domain name is bound.

Obtaining Domain Names

- To enable a service system on the cloud service platform to access APIs, obtain a private zone as an independent domain name.
  a. Apply for a private network domain name. For details, see "Private Zones" > "Creating a Private Zone" in the *Domain Name Service User Guide*
  b. Configure an A record set to map the domain name to the APIC address. For details, see "Adding an A Record Set" in the *Domain Name Service User Guide*
  c. If the API group contains HTTPS-compatible APIs, add an SSL certificate for the independent domain name bound to the group. Obtain the content and keys of the SSL certificate and **create an SSL certificate** in advance.
- To enable a service system outside the cloud service platform to access APIs, obtain a public zone as an independent domain name.
  a. Apply for a public zone from Domain Registration.
  b. Configure a CNAME record set for the API group subdomain name. For details, see "Adding a CNAME Record Set" in the *Domain Name Service User Guide*.
  c. If the API group contains HTTPS-compatible APIs, add an SSL certificate for the independent domain name bound to the group. Obtain the content and keys of the SSL certificate and **create an SSL certificate** in advance.

Procedure

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 Choose **API Management** > **API Groups**.

Step 4 Click a group name.

Step 5 Click the **Group Information** tab.

Step 6 In the **Independent Subdomain Names** area, click **Bind Independent Domain Name**. Then configure the domain name information.

Table 5-4 Independent domain name configuration

Parameter	Description
Domain Name	Domain name to be bound to the API group.
Minimum TLS Version	The minimum TLS version that can be used to access the domain name. TLS 1.1 and TLS 1.2 (recommended) are supported. This parameter applies only to HTTPS and does not take effect for HTTP and other access modes. Configure HTTPS cipher suites using the **ssl_ciphers** parameter on the **Parameters** tab.
HTTP-to-HTTPS Auto Redirection	Whether to support HTTP-to-HTTPS redirection. Redirection takes effect only when the API request protocol is **HTTPS** or **HTTP&HTTPS** and an SSL certificate has been bound to the independent domain name. **NOTE** Redirection is only suitable for GET and HEAD requests. Redirecting other requests may cause data loss due to browser restrictions.

Step 7 Click **OK**.

If the domain name is no longer needed, click **Unbind Domain Name** to unbind it from the API group.

Step 8 (Optional) If the API group contains HTTPS APIs, bind an SSL certificate to the independent domain name.

1. In the row that contains the domain name, click **Select SSL Certificate**.
2. Select an SSL certificate and click **OK**.
   - If a CA certificate has been uploaded for the SSL certificate, you can enable client authentication (HTTPS two-way authentication). **Enabling or disabling client authentication will affect the existing services. Exercise caution when performing this operation.**
   - If no SSL certificate is available, click **Create SSL Certificate** to create one. For details, see **SSL Certificates**.

----End

Troubleshooting

- Failure in binding an independent domain name: It already exists or is not CNAMEd to the debugging domain name of the API group.
- Failure in binding an SSL certificate: The domain name used to generate the SSL certificate is different from the target independent domain name.

Follow-Up Operations

After binding independent domain names to the API group, create APIs in the group to selectively expose backend capabilities. For details, see **Creating an API**.

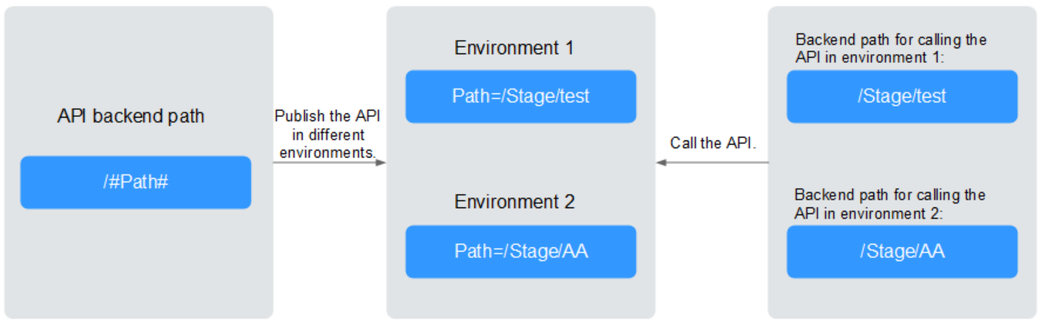
5.4 Creating an Environment Variable

You can define environment variables to allow an API to be called in different environments.

Environment variables are manageable and specific to environments. You can add variables in different environments to call different backend services using the same API.

For variables you define during API creation, you must create corresponding variables and values. For example, variable **Path** is defined for an API, and two variables with the same name are created and assigned values **/Stage/test** and **/Stage/AA** in environments 1 and 2, respectively. If the API is published and called in environment 1, the path **/Stage/test** is used. If the API is published and called in environment 2, the path **/Stage/AA** is used.

Figure 5-1 Use of environment variables


API backend path
/#Path#
Publish the API in different environments.
Environment 1
Path=/Stage/test
Environment 2
Path=/Stage/AA
Call the API.
Backend path for calling the API in environment 1:
/Stage/test
Backend path for calling the API in environment 2:
/Stage/AA

Procedure

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 Choose **API Management** > **API Groups**.

Step 4 Click a group name.

Step 5 Click the **Group Information** tab.

Step 6 In the **Environment Variables** area, select an environment. If no environment is available, click **Create Environment** to create one.

Step 7 Click **Add Environment Variable** and enter the variable information.

NOTICE

Environment variable names and values will be displayed in plain text in API requests. Do not include sensitive information in the variable names and values.

Table 5-5 Adding an environment variable

Parameter	Description
Name	Variable name. Ensure that the name is the same as the name of the variable defined for the API.
Value	The path to be used in the selected environment.

Step 8 Click **OK**.

----End

Follow-Up Operations

After creating an environment variable, you can **publish the API in the environment where the variable is located** so that the API can be called.

5.5 Creating a Gateway Response

A gateway response is displayed if APIG fails to process an API request. APIG provides a set of default responses and also allows you to create responses with custom status codes and content. The response content must be in JSON format.

For example, the content of a default gateway response is as follows:

```
{"error_code": "$context.error.code", "error_msg": "$context.error.message", "request_id":
"$context.requestId"}
```

You can add a response with the following content:

```
{"errorcode": "$context.error.code", "errormsg": "$context.error.message", "requestid":
"$context.requestId","apiId":"$context.apiId"}
```

You can add more fields to or delete existing fields from the JSON body.

NOTE

- You can create a maximum of four gateway responses for each group.
- A maximum of 10 response headers can be customized. The key of a response header can contain 1 to 128 characters, including digits, letters, and underscores (_). The value can reference runtime variables (see **Context Variables**), but cannot contain double brackets ([[or]]).
- The type of a default or custom response cannot be modified, but the status code and content of the response can.
- The type of a gateway response cannot be changed. For details, see **Response Types**.
- Gateway responses can contain the API gateway context variables (starting with **$context**). For details, see **Context Variables**.

Procedure

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 Choose **API Management** > **API Groups**.

Step 4 Click a group name.

Step 5 Click the **Group Information** tab.

Step 6 In the **Gateway Responses** area, create or modify gateway responses.

To cancel modifications to a default response, click **Restore Defaults** in the upper right.

----End

Response Types

The following table lists the response types supported by APIG. You can define status codes to meet your service requirements.

Table 5-6 Error response types supported by APIG

Response Name	Default Status Code	Description
Access Denied	403	Access denied. For example, the access control policy is triggered or an attack is detected.
Authorizer Configuration Error	500	A custom authorizer error has occurred. For example, communication failed or an error response was returned.
Authorizer Failed	500	The custom authorization failed.
Incorrect Identity Source	401	The identity source of the custom authorizer is missing or invalid.
Third-Party Configuration Error	500	A third-party authorizer error has occurred. For example, communication failed or an error response was returned.
Third-Party Authorizer Failure	401	The third-party authorizer returns an authentication failure.
Incorrect Third-Party Identity Source	401	The identity source of the third-party authorizer is missing.
Authentication Failure	401	IAM or app authentication failed.
Identity Source Not Found	401	No identity source has been specified.
Backend Timeout	504	Communication with the backend service timed out.
Backend Unavailable	502	The backend service is unavailable due to communication error.

Response Name	Default Status Code	Description
Default 4XX	-	Another 4XX error occurred.
Default 5XX	-	Another 5XX error occurred.
No API Found	404	No API is found.
Incorrect Request Parameters	400	The request parameters are incorrect or the HTTP method is not supported.
Request Throttled	429	The request was rejected due to request throttling.
Unauthorized Credential	401	The credential you are using has not been authorized to call the API.

Context Variables

Table 5-7 Variables that can be used in response message body

Variable	Description
$context.apiId	API ID.
$context.apiName	API name.
$context.appId	ID of the credential that calls the API.
$context.appName	Name of the credential that calls the API.
$context.requestId	Request ID generated when the API is called.
$context.stage	Deployment environment in which the API is called.
$context.sourceIp	Source IP address of the API caller.
$context.reqPath	API request path, excluding the query string.
$context.reqUri	API request path, including the query string.
$context.reqMethod	Request method.
$context.authorizer.frontend.property	Values of the specified attribute–value pairs mapped to the context in the frontend custom authorizer response
$context.authorizer.backend.property	Values of the specified attribute–value pairs mapped to the context in the backend custom authorizer response
$context.error.message	Error message.

Variable	Description
$context.error.code	Error code.
$context.error.type	Error type.

5.6 Creating an API

You can selectively expose your backends by configuring their APIs in APIG. To create an API, perform the following steps:

- **Configuring Frontend Settings**
  Frontend definitions, security settings, and request parameters
- **Configuring Backend Settings**
  Default backend, backend policies, and responses
- **(Optional) Creating a Policy**
  Traditional and plug-in policies

NOTE

APIG uses a REST-based API architecture, so API opening and calling must comply with related RESTful API specifications.

Prerequisites

- You have created an API group. If no API group is available, create one by referring to **Creating an API Group**.
- If the backend service needs to use a load balance channel, **create a channel** first.
- If you need to use a custom authorizer for API authentication, **create one**.

Configuring Frontend Settings

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 Choose **API Management** > **API Groups**.

Step 4 Click a group name.

Step 5 On the **APIs** tab, click **Create API** > **Create API**.

1. Configure the frontend parameters described in the following table.

   NOTE

   The new API must have a different group, request method, request path, and matching mode from those of any existing API.

Table 5-8 Frontend definition

Parameter	Description
API Name	Enter an API name that conforms to specific rules to facilitate search.
Group	The group to which the API belongs.
URL	Frontend address, which consists of a method, protocol, subdomain name, and path. – **Method**: Select **GET**, **POST**, **DELETE**, **PUT**, **PATCH**, **HEAD**, **OPTIONS**, or **ANY**. **ANY** indicates that the API can be called using any method. – **Protocol**: Select **HTTP**, **HTTPS**, or **HTTP&HTTPS**. **HTTPS** is recommended for transmitting important or sensitive data. APIG supports WebSocket data transmission. HTTP is equivalent to WebSocket (ws), and HTTPS is equivalent to WebSocket Secure (wss). – **Subdomain Name**: Debugging domain name of the group to which the API belongs. – **Path**: Request path of the API. Enclose parameters in braces, if any. For example: **/a/{b}**. Alternatively, use a plus sign (+) to match paths starting with specific characters. For example: **/a/{b+}**. The request path is case-sensitive.
Gateway Response	Displayed if an API request fails to be processed. APIG provides a set of default responses and also allows you to **create new ones** with custom status codes and content on the **Group Information** page. The response content must be in JSON format.

Parameter	Description
Matching	Matching mode of the API request path. – **Exact match**: The path in a request for the API must be the same as the value of **Path**. – **Prefix match**: The path in a request for the API must be prefixed with the value of **Path**. You can define multiple paths in this mode. For example, if you set **Path** to **/test/AA** and **Matching** to **Prefix match**, the API can be called using **/test/AA/BB** and **/test/AA/CC** but cannot be called using **/test/AACC**. **NOTE** – If you set the matching mode to **Prefix match**, the characters of the API request path excluding the prefix are transparently transmitted to the backend. For example, if you define the frontend and backend request paths of an API as **/test/** and **/test2/**, respectively, and the API is called using **/test/AA/CC**, the characters **AA/CC** will be transparently transmitted to the backend. The request URL received by the backend is **/test2/AA/CC/**. – In prefix match mode, the path in a request preferentially matches the API with the longest path. For example, assume that prefix match is enabled for two APIs whose paths are **/test/AA** and **/test/AA/BB**, respectively. The path **/test/AA/BB/c** in a request matches the API whose path is **/test/AA/BB**. – If there are two APIs with the same group, request method, and request path, the API with exact matching is first called.
Tags	Attributes used to quickly identify the API from other APIs.
Description	Description of the API.
Request Body Format	Enable the parameter to specify a format for API requests. APIG will transmit API requests to the backend by using the selected format. The options include **application/json**, **application/xml**, **text/plain**, and **multipart/form-data**. The selected format must be supported by the backend service.
Request Body Content	Enter the content of the request body in the API request to help API callers understand how to correctly encapsulate API requests.

Parameter	Description
Base64 Encoding	Enabled by default to Base64-encode the body of API requests for interacting with FunctionGraph. **Base64 encoding works only when any of the following conditions is met:** – **A custom authorizer is used.** – **The backend type is FunctionGraph.** – **A circuit breaker policy is bound, using FunctionGraph for backend downgrade.** You can disable Base64 encoding only when the content format is application/json.

2. Configure security settings based on the following table.

Table 5-9 Security configuration

Parameter	Description
Visibility	Determine whether the API is available to the public. Options: – **Public**: The API can be released to KooGallery. – **Private**: The API will be excluded when the API group to which it belongs is released on KooGallery.
Authentication Mode	The following authentication modes are available: – **App**: Requests for the API will be authenticated by APIG. App authentication is recommended. – **IAM**: Requests for the API will be authenticated by Identity and Access Management (IAM). – **Custom**: Requests for the API will be authenticated by using your own authentication system or service (for example, an OAuth-based authentication system). – **None**: No authentication will be required. API calling varies depending on the authentication mode. For details, see **Calling APIs**. **NOTICE** – If you set the authentication mode to **IAM** or **None**, any APIG user can access the API, which can result in excessive charges if the API is bombarded with malicious requests. – If you set the authentication mode to **Custom**, you can create a function in FunctionGraph to interconnect with your own authentication system or service. Ensure that **FunctionGraph** is available in the current region.

Parameter	Description
Simple Authentication	This parameter is available only if you set **Security Authentication** to **App**. If you select app authentication, configure whether to enable simple authentication. In simple authentication, the **X-Apig-AppCode** parameter is added to the HTTP request header for quick response. APIG verifies only the AppCode and the request content does not need to be signed. Simple authentication supports HTTPS and gRPCS requests and does not support HTTP requests. For details, see **Adding an AppCode for Simple Authentication**. **NOTE** After you enable simple authentication for an existing API, you need to publish the API again. For details, see **Publishing an API**.
Two-Factor Authentication	This parameter is available only if **Authentication Mode** is set to **App** or **IAM**. Determine whether to enable two-factor authentication for the API. If this option is enabled, API requests will be authenticated using a custom authorizer in addition to the app or IAM authentication you specify.
Custom Authorizer	This parameter is mandatory only if **Authentication Mode** is set to **Custom**. If no custom authorizer is available, click **Create Custom Authorizer** to create one.
CORS	Determine whether to enable cross-origin resource sharing (CORS). CORS allows browsers to send XMLHttpRequest to servers in other domains, overcoming the limitation that Asynchronous JavaScript and XML (AJAX) can be used only within the same domain. There are two types of CORS requests: – Simple requests: requests that have the **Origin** field in the header. – Not-so-simple requests: HTTP requests sent before the actual request. If CORS (not-so-simple request) is enabled for an API, another API that uses the OPTIONS method must be created. For details, see **Enabling CORS**.

3. (Optional) Define request parameters described in the following table.

Table 5-10 Request parameter configuration

Parameter	Description
Parameter Name	Request parameter name. The name of a path parameter will be automatically displayed in this column. **NOTE** – The parameter name is case-insensitive. It cannot start with **x-apig-** or **x-sdk-**. – The parameter name cannot be **x-stage**. – If you set the parameter location to **HEADER**, ensure that the parameter name is not **Authorization** or **X-Auth-Token** and does not contain underscores (_).
Parameter Type	Options: **STRING** and **NUMBER**. **NOTE** Set the type of Boolean parameters to **STRING**.
Required	Determine whether the parameter is required in each request sent to call the API. If you select **Yes**, API requests that do not contain the parameter will be rejected.
Passthrough	Determine whether to transparently transmit the parameter to the backend service.
Enumerated Value	Enumerated value of the parameter. Use commas (,) to separate multiple enumerated values. The value of this parameter can only be one of the enumerated values.
Default Value	The value that will be used if no value is specified for the parameter when the API is called. If the parameter is not specified in a request, APIG automatically sends the default value to the backend service.
Value Restrictions	– Max. length/Max. value: If **Parameter Type** is set to **STRING**, set the maximum length of the parameter value. If **Parameter Type** is set to **NUMBER**, set the maximum parameter value. – Min. length/Min. value: If **Parameter Type** is set to **STRING**, set the minimum length of the parameter value. If **Parameter Type** is set to **NUMBER**, set the minimum parameter value.
Example	Example value for the parameter.
Description	Description of the parameter.

Step 6 Click **Next** to proceed with **Configuring Backend Settings**.

----End

Configuring Backend Settings

APIG allows you to define multiple backend policies for different scenarios. Requests that meet specified conditions will be forwarded to the corresponding backend. For example, you can have certain requests to an API forwarded to a specific backend by specifying the source IP address in the policy conditions of the backend.

You can define a maximum of five backend policies for an API in addition to the default backend.

Step 1 Define the default backend.

API requests that do not meet the conditions of any backend will be forwarded to the default backend.

On the **Backend Configuration** page, select a backend type.

APIG supports **HTTP&HTTPS, FunctionGraph**, and **Mock** backends. For details about the parameters required for defining each type of backend service, see **Table 5-11**, **Table 5-12**, and **Table 5-13**.

NOTE

- FunctionGraph backends can be set only if FunctionGraph has been deployed in the current environment.
- If the backend service is unavailable, use the Mock mode to return the expected result to the API caller for debugging and verification.

Table 5-11 Parameters for defining an HTTP&HTTPS backend service

Parameter	Description
Load Balance Channel	Determine whether to use a load balance channel to access the backend service. If you select **Configure**, ensure that you have **created a load balance channel**.

Parameter	Description
URL	A URL consists of a method, protocol, load balance channel/backend address, and path. ● Method Select **GET**, **POST**, **DELETE**, **PUT**, **PATCH**, **HEAD**, **OPTIONS**, or **ANY**. **ANY** indicates that all request methods are supported. ● Protocol **HTTP** or **HTTPS**. **HTTPS** is recommended for transmitting important or sensitive data. **NOTE** – APIG supports WebSocket data transmission. HTTP is equivalent to WebSocket (ws), and HTTPS is equivalent to WebSocket Secure (wss). – This protocol must be the one used by the backend service. ● Load Balance Channel (if applicable) Select a load balance channel. **NOTE** To ensure a successful health check and service availability, configure the security groups of cloud servers in each channel to allow access from 100.125.0.0/16. ● Backend Address (if applicable) **Set this parameter if no load balance channel is used.** Enter the access address of the backend service in the format of *Host:Port*. *Host* indicates the IP address or domain name for accessing the backend service. If no port is specified, port 80 is used for HTTP by default, and port 443 is used for HTTPS by default. To use environment variables in the backend address, enclose the variables with number signs (#), for example, **#ipaddress#**. You can use multiple environment variables, for example, **#ipaddress##test#**. ● Path The request path (URI) of the backend service. Ensure that any parameters in the path are enclosed in braces ({}). For example, **/getUserInfo/***{userId}*. If the path contains an environment variable, enclose the environment variable in number signs (#), for example, **/#path#**. You can use multiple environment variables, for example, **/#path##request#**.

Parameter	Description
Host Header (if applicable)	**Set this parameter only if a load balance channel is used.** Define a host header for requests to be sent to cloud servers associated with the load balance channel. By default, the original host header in each request is used.
Timeout (ms)	Backend request timeout. Range: 1–60,000 ms. If a backend timeout error occurs during API debugging, increase the timeout to locate the reason. **NOTE** If the current timeout does not meet your service requirements, modify the maximum timeout by referring to **Configuring Parameters**. The value range is 1 ms to 600,000 ms. After modifying the maximum timeout, also modify the timeout here.
Retries	Number of attempts to retry requesting the backend service. Default: **0**; range: –1 to 10. ● If the value is **–1**, the retry function is disabled. However, requests except for those using **POST** and **PATCH** will be retried once by default. ● If the value is within 0 to 10, the retry function is enabled, and requests will retry for the specified number of times. **0** indicates no retry attempts will be made. If a load balance channel is used, the number of retries must be less than the number of enabled backend servers in the channel.
Two-Way Authentication	**Set this parameter only when Protocol is set to HTTPS.** Determine whether to enable two-way authentication between APIG and the backend service. If you enable this option, configure the **backend_client_certificate** parameter on the **Parameters** page of the gateway.
Backend Authentication	Determine whether your backend service needs to authenticate API requests. If you enable this option, select a custom authorizer for backend authentication. **Custom authorizers** are functions that are created in FunctionGraph to implement an authentication logic or to invoke an authentication service. **NOTE** Backend authentication relies on FunctionGraph and is only available in certain regions.

Table 5-12 Parameters for defining a FunctionGraph backend service

Parameter	Description
Function Name	Automatically displayed when you select a function.
Function URN	Identifier of the function. Click **Select** to specify a function.
Version/Alias	Select a version or alias for your function. For details, see "Version Management" in the *FunctionGraph User Guide* or "Alias Management" in the *FunctionGraph User Guide*.
Invocation Mode	• **Synchronous**: When receiving an invocation request, FunctionGraph immediately processes the request and returns a result. The client closes the connection once it has received a response from the backend. • **Asynchronous**: The function invocation results of client requests do not matter to clients. When it receives a request, FunctionGraph queues the request, returns a response, and then processes requests one by one in idle state.
Timeout (ms)	Timeout duration for APIG to request for the backend service. For details, see the description about backend timeout in **Table 5-11**.
Backend Authentication	For details, see the description about backend authentication in **Table 5-11**.

Table 5-13 Parameters for defining a Mock backend service

Parameter	Description
Status Code	Select the HTTP status code to be returned by the API.
Response	You can use Mock for API development, debugging, and verification. It enables APIG to return a response without sending the request to the backend. This is useful if you need to test APIs when the backend is unavailable.
Backend Authentication	For details, see the description about backend authentication in **Table 5-11**.
Add Header	Customize the response header parameters for the API. Click **Add Header**, and enter the parameter name, value, and description.

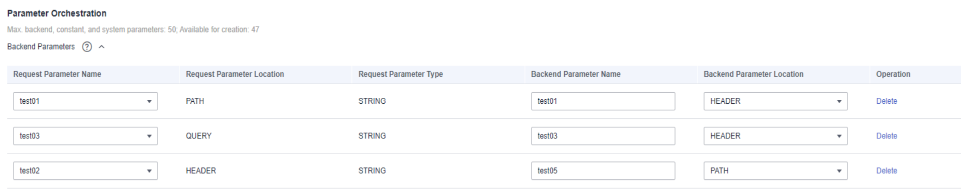
NOTE

- APIs whose URLs contain variables cannot be debugged on the API debugging page.
- For variables defined in URLs of APIs, corresponding environment variables and their values must be configured. Otherwise, the APIs cannot be published because there will be no values that can be assigned to the variables.
- The variable name is case-sensitive.

Step 2 (Optional) Configure backend parameters to map them to the request parameters defined in corresponding locations. If no request parameter is defined in 5.3, skip this step.

1. In the **Backend Parameters** area, add parameters in either of the following ways:
   – Click **Import Request Parameter** to synchronize all defined request parameters.
   – Click **Add Backend Parameter Mapping** to add a backend parameter.
2. Modify mappings (see Figure 5-2) based on the parameters and their locations in backend requests.

Figure 5-2 Configuring backend parameters


Parameter Orchestration

Max. backend, constant, and system parameters: 50; Available for creation: 47

Backend Parameters

Request Parameter Name	Request Parameter Location	Request Parameter Type	Backend Parameter Name	Backend Parameter Location	Operation
test01	PATH	STRING	test01	HEADER	Delete
test03	QUERY	STRING	test03	HEADER	Delete
test02	HEADER	STRING	test05	PATH	Delete

a. If the parameter location is set to **PATH**, the parameter name must be the same as that defined in the backend request path.

b. The name and location of a request parameter can be different from those of the mapped backend parameter.

NOTE

- The parameter name is case-insensitive. It cannot start with **x-apig-** or **x-sdk-**.
- The parameter name cannot be **x-stage**.
- If you set the parameter location to **HEADER**, ensure that the parameter name does not start with an underscore (_).

c. In the preceding figure, parameters **test01** and **test03** are located in the path and query positions of API requests, and their values will be received in the header of backend requests. **test02** is located in the header of API requests, and its value will be received through **test05** in the path of backend requests.

Assume that **test01** is **aaa**, **test02** is **bbb**, and **test03** is **ccc**.

The API request is as follows:

```
curl -ik -H 'test02:bbb' -X GET https://example.com/v1.0/aaa?test03=ccc
```

Backend request:

```
curl -ik -H 'test01:aaa' -H 'test03:ccc' -X GET https://example.com/v1.0/bbb
```

Step 3 (Optional) Configure constant parameters for the default backend to receive constants that are invisible to API callers. When sending a request to the backend service, APIG adds these parameters to the specified locations in the request and then sends the request to the backend service.

In the **Constant Parameters** area, click **Add Constant Parameter**.

NOTICE

Constant parameters will be stored as plaintext. To prevent information leakage, do not contain sensitive information in these parameters.

Table 5-14 Constant parameter configuration

Parameter	Description
Constant Parameter Name	If **Parameter Location** is set to **PATH**, the parameter name must be the same as that in **Path**. **NOTE** ● The parameter name is case-insensitive. It cannot be **x-stage** or start with **x-apig-** or **x-sdk-** ● If **Parameter Location** is set to **HEADER**, the parameter name is case-insensitive and cannot start with an underscore (_).
Parameter Location	Specify the location of the constant parameter in backend service requests. The options include **PATH**, **HEADER**, and **QUERY**.
Parameter Value	Value of the constant parameter.
Description	Description about the constant parameter.

NOTE

- APIG sends requests containing constant parameters to a backend service after percent-encoding of special parameter values. Ensure that the backend service supports percent-encoding. For example, parameter value **[api]** becomes **%5Bapi%5D** after percent-encoding.
- For values of path parameters, APIG percent-encodes the following characters: ASCII codes 0–31 and 127–255, spaces, and other special characters ?></%#"[\]^`{|}
- For values of query strings, APIG percent-encodes the following characters: ASCII codes 0–31 and 127–255, spaces, and other special characters >=<+&%#"[\]^`{|}

Step 4 (Optional) Configure system parameters for the default backend to receive default gateway parameters, frontend authentication parameters, and backend authentication parameters. When sending a request to the backend service, APIG adds these parameters to the specified locations in the request and then sends the request to the backend service.

1. In the **System Parameters** area, click **Add System Parameter**.

Table 5-15 System parameter configuration

Parameter	Description
System Parameter Type	Options: – **Default gateway parameter**: Parameters supported by APIG. – **Frontend authentication parameter**: Parameters to be displayed in the frontend custom authentication result. This option is available only if you have set **Authentication Mode** to **Custom** or enabled **Two-Factor Authentication** in **Configuring Frontend Settings**. – **Backend authentication parameter**: Parameters to be displayed in the backend custom authentication result. This option is available only if you have enabled backend authentication in **Configuring Backend Settings**.

Parameter	Description
System Parameter Name	Name of the system parameter. – If **System Parameter Type** is **Default gateway parameter**, select any of the following parameters. ■ **sourceIp**: source IP address of an API caller ■ **stage**: environment in which the API is called ■ **apiId**: ID of the API ■ **appId**: ID of the app that calls the API ■ **requestId**: request ID generated when the API is called ■ **serverAddr**: IP address of the gateway server ■ **serverName**: name of the gateway server ■ **handleTime**: processing time of the API request ■ **providerAppId**: credential ID of the API provider ■ **apiName**: name of the API. This parameter is available only after the API is published. ■ **appName**: name of the credential used to call the API – If **System Parameter Type** is **Frontend authentication parameter** or **Backend authentication parameter**, enter a parameter that has been defined for custom authentication results. For details about how to create a custom authorizer function and obtain result parameters, see *API Gateway Developer Guide*.
Backend Parameter Name	Name of a backend parameter to map the system parameter. **NOTE** – The parameter name is case-insensitive. It cannot be **x-stage** or start with **x-apig-** or **x-sdk-** – If **Parameter Location** is set to **HEADER**, the parameter name is case-insensitive and cannot start with an underscore (_).
Backend Parameter Location	Specify the location of the backend parameter in backend service requests. The options include **PATH**, **HEADER**, and **QUERY**.

Parameter	Description
Description	Description about the system parameter.

Step 5 (Optional) Add a backend policy.

You can add backend policies to forward requests to different backend services.

1. Click ⊕ to add a backend policy.
2. Set policy parameters described in **Table 5-16**. For details about other parameters, see **Table 5-11**, **Table 5-12**, and **Table 5-13**.

Table 5-16 Backend policy parameters

Parameter	Description
Name	The backend policy name.
Effective Mode	– **Any condition met**: The backend policy takes effect if any of the policy conditions has been met. – **All conditions met**: The backend policy takes effect only when all the policy conditions have been met.
Policy Conditions	Conditions that must be met for the backend policy to take effect. Set conditions by referring to **Table 5-17**.

Table 5-17 Policy condition configuration

Parameter	Description
Source	– **Source IP address**: IP address from which the API is called – **Request parameter**: a request parameter defined for the API – **Cookie**: cookies of an API request – **System parameter - Default gateway parameter**: a default gateway parameter used to define system runtime for the API – **System parameter - Frontend authentication parameter**: displayed in the frontend custom authentication result This option is available only if you have set **Authentication Mode** to **Custom** or enabled **Two-Factor Authentication** in **Configuring Frontend Settings**. NOTICE – The request parameters (for example, headers) set as policy conditions must have already been defined for the API. – If **System parameter** is not displayed, contact technical support to upgrade the gateway.
Parameter Name	– When setting **Source** to **Request parameter**, select a request parameter. – When setting **Source** to **System parameter**, select a system parameter. ▪ **reqPath**: Request URI, for example, **/a/b/c**. ▪ **reqMethod**: Request method, for example, **GET**. – When setting **Source** to **Cookie**, enter the name of a cookie parameter.
Parameter Location	The parameter location is displayed only if you set **Source** to **Request parameter**.
Condition Type	This parameter is required only if you set **Source** to **Request parameter**, **System parameter**, or **Cookie**. – **Equal**: The request parameter must be equal to the specified value. – **Enumerated**: The request parameter must be equal to any of the enumerated values. – **Matching**: The request parameter must be equal to any value of the regular expression. NOTE When you set **Source** to **System parameter** and select a parameter named **reqMethod**, you can set the condition type only to **Equal** or **Enumerated**.

Parameter	Description
Condition Value	– If **Condition Type** is **Equal**, enter a value. – If **Condition Type** is **Enumerated**, enter multiple values and separate them with commas (,). – If **Condition Type** is **Matching**, enter a value range, for example, **[0-5]**. – If **Source** is **Source IP address**, enter one or more IP addresses and separate them with commas (,). – If **Source** is **System parameter - Frontend authentication parameter** and the condition value is of the Boolean type, the parameter must be in lowercase.

Step 6 Defining responses.

In the **Responses** area, set the example responses.

Table 5-18 Defining responses

Parameter	Description
Example Success Response	The response to be returned when the API is called successfully.
Example Failure Response	The response to be returned when the API fails to be called.

Step 7 Click **Finish**. You can view the API details on the **APIs** tab that is displayed.

----End

(Optional) Creating a Policy

You can create policies for the API after publishing it.

Step 1 On the **APIs** tab, click **Create Policy**.

Step 2 Select a policy type and set parameters.

- Select existing policy
- Create new policy (see **Creating a Policy and Binding It to APIs**)

Step 3 Click **OK**.

----End

FAQs About API Creation

Does APIG Support Multiple Backend Endpoints?

What Are the Possible Causes If the Message "Backend unavailable" or "Backend timeout" Is Displayed?

Why Am I Seeing the Message "No backend available"?

Follow-Up Operations

After creating an API, verify it by following the procedure in **Debugging an API**.

5.7 Creating a gRPC API

APIG supports gRPC API creation. gRPC is a modern, open-source, high-performance Remote Procedure Call (RPC) framework that can run in any environment. You only need to define the request and response of each API, and let the gRPC framework take care of the rest. gRPC uses protocol buffers (protobuf) as its Interface Definition Language (IDL) and for bottom-layer message exchange. The following table compares gRPC and REST APIs.

Table 5-19 gRPC vs REST

Parameter	gRPC	REST
Message encoding	protobuf	JSON
Transmission protocol	HTTP/2	HTTP
Transmission performance	Fast, with less content to transmit	More content to transmit

Parameter	gRPC	REST
Transmission mode	• Unary RPC Send a single request and receive a single response. • Server streaming RPC Send a single request and receive multiple responses. • Client streaming RPC Send multiple requests and receive a single response. • Bidirectional streaming RPC Send multiple requests and receive multiple responses.	Send a single request and receive a single response.

If both your client and server are of the gRPC type, you can create an gRPC API to open up your backend capabilities. gRPC features low resource consumption and high transmission rate. It is suitable for internal service invocation and governance.

Restrictions

- gRPC APIs cannot be imported, exported, or debugged, and do not support the import of API design files, CSE microservices, or CCE workloads.
- Circuit breaker policies whose backend policy type is **Mock**, **HTTP&HTTPS**, or **FunctionGraph** are not supported.

Prerequisites

- You have created an API group. If no API group is available, create one by referring to **Creating an API Group**.
- If the backend service needs to use a load balance channel, **create a channel** first.
- The backend service has a proto file that defines API request and response parameters. The proto file is used in gRPC to define data structures and service APIs. It describes data structures and interactions using protobuf and serves as a contract for communication between the client and backend services.

Procedure

Step 1 Go to the APIG console.

Step 2 Select a gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **API Management** > **API Groups**.

Step 4 Click a group name.

Step 5 On the **APIs** tab page, choose **Create API** > **Create gRPC API**.

Step 6 Configure the frontend definition according to **5.1**.

For gRPC APIs, the default frontend request method is **POST** and the protocol is **GRPCS**.

Set the path to any of the following:

- /
- /*{Package name}*.*{Service name}*
- /*{Package name}*.*{Service name}*/*{Method name}*

NOTE

- Obtain the package name, service name, and method name from the **proto file**.
- Absolute match can be used only when the frontend path is set to **/***{Package name}***.** *{Service name}***/***{Method name}*.

Step 7 Configure the authentication mode by referring to **5.2**.

Step 8 Click **Next**.

Step 9 Configure the default backend by referring to **Step 1**.

The backend service type of gRPC APIs can be **GRPC&GRPCS** or **FunctionGraph**.

- When the type is **GRPC&GRPCS**, the backend service uses the **POST** request method, **/** path, and **GRPC** or **GRPCS** protocol, and does not support parameter orchestration.
- When the type is FunctionGraph, the backend service uses **V2** network architecture and **Synchronous** invocation type by default, and does not support parameter orchestration.

Step 10 (Optional) Add a backend policy by referring to **Step 5**.

----End

(Optional) Creating a Policy

You can create policies for the API after publishing it.

Step 1 On the **APIs** tab, click **Create Policy**.

Step 2 Select a policy type and set parameters.

- Select existing policy
- Create new policy (see **Creating a Policy and Binding It to APIs**)

Step 3 Click **OK**.

----End

5.8 Cloning an API

To improve API creation efficiency, you can clone an API with a custom name and path.

Policies bound to an API cannot be cloned and can only be manually bound to the new API.

Prerequisites

You have created an API. If no API is available, create one by referring to **Creating an API**.

Procedure

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **API Management** > **API Groups**.

Step 4 Click a group name.

Step 5 On the **APIs** tab, choose **More** > **Clone**.

Step 6 Set the API name and path, and click **OK**.

----End

Follow-Up Operations

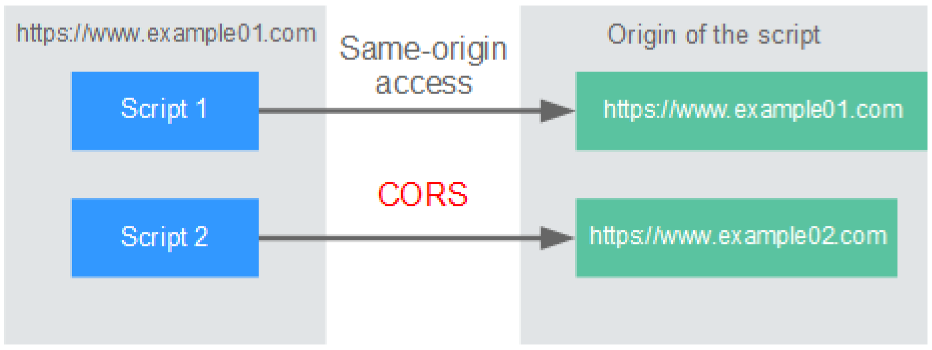
After cloning an API, verify it by following the procedure in **Debugging an API**.

5.9 CORS

What Is CORS?

For security reasons, browsers restrict cross-origin requests initiated from within scripts. This means that a web application can only request resources from its origin. The CORS mechanism allows browsers to send XMLHttpRequest to servers in other domains and request access to the resources there.

Figure 5-3 Process flow of the CORS mechanism


https://www.example01.com
Script 1
Script 2
Same-origin access
CORS
Origin of the script
https://www.example01.com
https://www.example02.com

There are two types of CORS requests:

- **Simple requests**

  Simple requests must meet the following conditions:

  a. The request method is HEAD, GET, or POST.

b. The request header contains only the following fields:
   - Accept
   - Accept-Language
   - Content-Language
   - Last-Event-ID
   - Content-Type (**application/x-www-form-urlencoded**, **multipart/form-data**, or **text/plain**)

   In the header of a simple request, browsers automatically add the **Origin** field to specify the origin (including the protocol, domain, and port) of the request. After receiving such a request, the target server determines whether the request is safe and can be accepted based on the origin. If the server sends a response containing the **Access-Control-Allow-Origin** field, the server accepts the request.

- **Not-so-simple requests**

  Requests that do not meet the conditions for simple requests are not-so-simple requests.

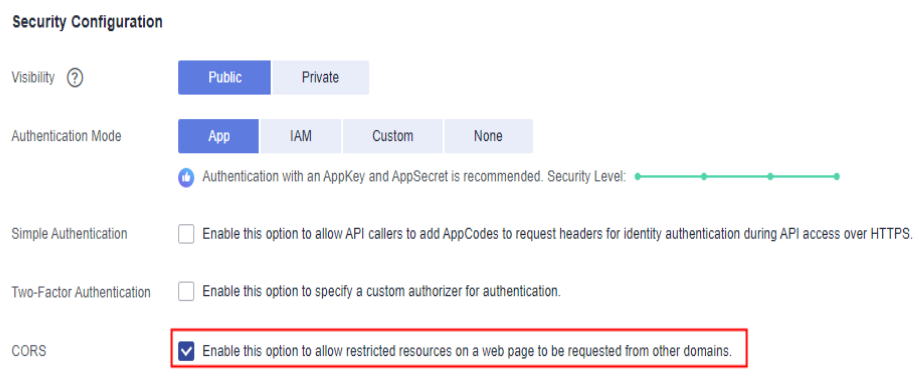
  Before sending a not-so-simple request, browsers send an HTTP preflight request to the target server to confirm whether the origin the web page is loaded from is in the allowed origin list, and to confirm which HTTP request methods and header fields can be used. If the preflight request is successful, browsers send simple requests to the server.

Configuring CORS

CORS is disabled by default. To enable CORS for an API, perform the operations described in this section. To customize request headers, request methods, and origins allowed for cross-domain access, create a CORS plug-in policy by referring to **CORS**.

- **Simple CORS requests**

  When creating an API, enable CORS in the **Security Configuration** area of the **Create API** page. For more information, see **Simple Request**.



Security Configuration

Visibility: [✓ Public] Private

Authentication Mode: [✓ App] IAM Custom None

Authentication with an AppKey and AppSecret is recommended. Security Level:

Simple Authentication: ☐ Enable this option to allow API callers to add AppCodes to request headers for identity authentication during API access over HTTPS.

Two-Factor Authentication: ☐ Enable this option to specify a custom authorizer for authentication.

CORS: ☑ Enable this option to allow restricted resources on a web page to be requested from other domains.

- **Not-so-simple CORS requests**

NOTICE

Not-so-simple CORS requests can be implemented in either of the following ways:

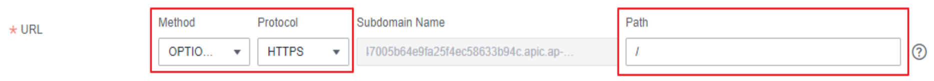
Method 1: Create an API that uses the **OPTIONS** method for preflight. Follow this procedure to define the preflight request API. For details, see **Not-So-Simple Request**.

Method 2: Configure a CORS policy and bind it to the API. For details, see **CORS**.

a. In the **Frontend Definition** area, set the following parameters:

- **Method**: Select **OPTIONS**.
- **Protocol**: The same protocol used by the API with CORS enabled.
- **Path**: Enter a slash (/).

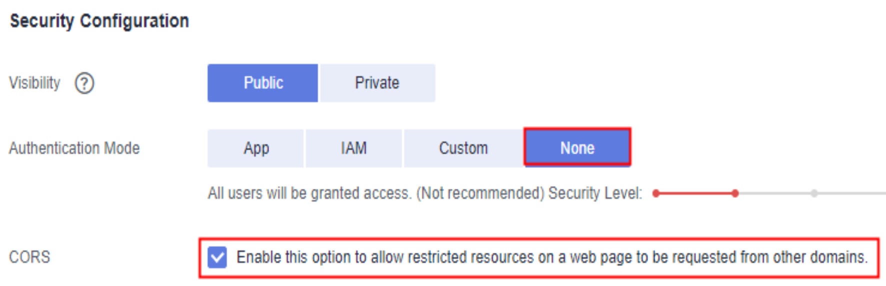
Figure 5-4 Defining the API request



* URL
Method: OPTIO...
Protocol: HTTPS
Subdomain Name: I7005b64e9fa25f4ec58633b94c.apic.ap-...
Path: /

b. In the **Security Configuration** area, select **None** and enable **CORS**.

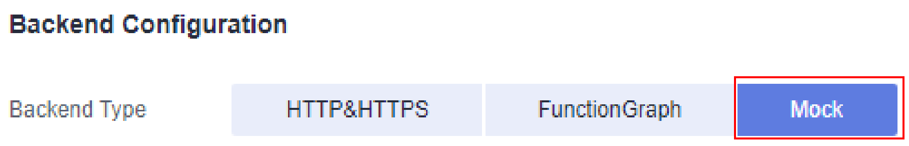
Figure 5-5 None authentication



Security Configuration
Visibility: Public | Private
Authentication Mode: App | IAM | Custom | None
All users will be granted access. (Not recommended) Security Level:
CORS: ✓ Enable this option to allow restricted resources on a web page to be requested from other domains.

c. Select the **Mock** backend type.

Figure 5-6 Mock backend service



Backend Configuration
Backend Type: HTTP&HTTPS | FunctionGraph | Mock

Simple Request

When creating an API that will receive simple requests, **enable CORS** for the API.

Scenario 1: If CORS is enabled and the response from the backend does not contain a CORS header, APIG handles requests from any domain, and returns the **Access-Control-Allow-Origin** header. For example:

Request sent by a browser and containing the Origin header field:

```
GET /simple HTTP/1.1
Host: www.test.com
Origin: http://www.cors.com
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Accept: application/json
Date: Tue, 15 Jan 2019 01:25:52 GMT
```

Origin: This field is required to specify the origin (**http://www.cors.com** in this example) of the request. APIG and the backend service determine based on the origin whether the request is safe and can be accepted.

Response sent by the backend service:

```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 01:25:52 GMT
Content-Type: application/json
Content-Length: 16
Server: api-gateway

{"status":"200"}
```

Response sent by APIG:

```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 01:25:52 GMT
Content-Type: application/json
Content-Length: 16
Server: api-gateway
X-Request-Id: 454d689fa69847610b3ca486458fb08b
Access-Control-Allow-Origin: *

{"status":"200"}
```

Access-Control-Allow-Origin: This field is required. The asterisk (*) means that APIG handles requests sent from any domain.

Scenario 2: If CORS is enabled and the response from the backend contains a CORS header, the header will overwrite that added by APIG. The following messages are used as examples:

Request sent by a browser and containing the Origin header field:

```
GET /simple HTTP/1.1
Host: www.test.com
Origin: http://www.cors.com
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Accept: application/json
Date: Tue, 15 Jan 2019 01:25:52 GMT
```

Origin: This field is required to specify the origin (**http://www.cors.com** in this example) of the request. APIG and the backend service determine based on the origin whether the request is safe and can be accepted.

Response sent by the backend service:

```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 01:25:52 GMT
Content-Type: application/json
Content-Length: 16
Server: api-gateway
Access-Control-Allow-Origin: http://www.cors.com

{"status":"200"}
```

Access-Control-Allow-Origin: Indicates that the backend service accepts requests sent from **http://www.cors.com**.

Response sent by APIG:

```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 01:25:52 GMT
Content-Type: application/json
Content-Length: 16
Server: api-gateway
X-Request-Id: 454d689fa69847610b3ca486458fb08b
Access-Control-Allow-Origin: http://www.cors.com

{"status":"200"}
```

The CORS header in the backend response overwrites that in APIG's response.

Not-So-Simple Request

When creating an API that will receive not-so-simple requests, enable CORS for the API by following the instructions in **Configuring CORS**, and create another API that will be accessed using the OPTIONS method.

The request parameters of an API accessed using the OPTIONS method must be set as follows:

- **Group**: The same group to which the API with CORS enabled belongs.
- **Method**: Select **OPTIONS**.
- **Protocol**: The same protocol used by the API with CORS enabled.
- **Path**: Enter a slash (/) or select the path that has been set for or matches the API with CORS enabled.
- **Security Authentication**: Select **None**. No authentication is required for requests received by the new API no matter which security authentication mode has been selected.
- **CORS**: Enable this option.

The following are example requests and responses sent to or from a mock backend.

Request sent from a browser to an API that is accessed using the OPTIONS method:

```
OPTIONS /HTTP/1.1
User-Agent: curl/7.29.0
Host: localhost
Accept: */*
Origin: http://www.cors.com
Access-Control-Request-Method: PUT
Access-Control-Request-Headers: X-Sdk-Date
```

- **Origin**: This field is required to specify the origin from which the request has been sent.
- **Access-Control-Request-Method**: This field is required to specify the HTTP methods to be used by the subsequent simple requests.
- **Access-Control-Request-Headers**: This field is optional and used to specify the additional header fields in the subsequent simple requests.

Response sent by the backend: none

Response sent by APIG:

```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 02:38:48 GMT
```

```
Content-Type: application/json
Content-Length: 1036
Server: api-gateway
X-Request-Id: c9b8926888c356d6a9581c5c10bb4d11
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: X-Stage,X-Sdk-Date,X-Sdk-Nonce,X-Proxy-Signed-Headers,X-Sdk-Content-Sha256,X-Forwarded-For,Authorization,Content-Type,Accept,Accept-Ranges,Cache-Control,Range
Access-Control-Expose-Headers: X-Request-Id,X-Apig-Latency,X-Apig-Upstream-Latency,X-Apig-RateLimit-Api,X-Apig-RateLimit-User,X-Apig-RateLimit-App,X-Apig-RateLimit-Ip,X-Apig-RateLimit-Api-Allenv
Access-Control-Allow-Methods: GET,POST,PUT,DELETE,HEAD,OPTIONS,PATCH
Access-Control-Max-Age: 172800
```

- **Access-Control-Allow-Origin**: This field is required. The asterisk (*) means that APIG handles requests sent from any domain.
- **Access-Control-Allow-Headers**: This field is required if it is contained in the request. It indicates all header fields that can be used during cross-origin access.
- **Access-Control-Expose-Headers**: This is the response header fields that can be viewed during cross-region access.
- **Access-Control-Allow-Methods**: This field is required to specify which HTTP request methods the APIG supports.
- **Access-Control-Max-Age**: This field is optional and used to specify the length of time (in seconds) during which the preflight result remains valid. No more preflight requests will be sent within the specified period.

Request sent by a browser and containing the Origin header field:

```
PUT /simple HTTP/1.1
Host: www.test.com
Origin: http://www.cors.com
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Accept: application/json
Date: Tue, 15 Jan 2019 01:25:52 GMT
```

Response sent by the backend:

```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 01:25:52 GMT
Content-Type: application/json
Content-Length: 16
Server: api-gateway

{"status":"200"}
```

Response sent by APIG:

```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 01:25:52 GMT
Content-Type: application/json
Content-Length: 16
Server: api-gateway
X-Request-Id: 454d689fa69847610b3ca486458fb08b
Access-Control-Allow-Origin: *

{"status":"200"}
```

5.10 Debugging an API

After creating an API, debug it on the APIG console by setting HTTP headers and body to verify whether the API is running normally.

NOTE

- APIs with backend request paths containing variables cannot be debugged.
- If a **plug-in or traditional policy** is bound to an API, the policy does not take effect during API debugging.
- The maximum backend timeout is 60s for API debugging.

Prerequisites

You have set up the backend service of the API.

Procedure

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 Choose **API Management** > **API Groups**.

Step 4 Click a group name.

Step 5 On the **APIs** tab, select the target API and click **Debug**.

Step 6 Configure the URL and request parameters of the API.

Select a request method, protocol, and domain name, and set request parameters.

Select the debugging or an independent domain name. If you select a wildcard domain name, specify the subdomain name.

NOTE

If the independent domain name you select is a wildcard domain name, you can use any of its subdomain names to access all APIs in the group to which the domain name is bound.

For example, if a wildcard domain name is ***.aaa.com**, the subdomain name can be **default.aaa.com** or **1.aaa.com**.

Step 7 Click **Debug**.

Step 8 The box on the lower right displays the response of the API request.

- If the debugging is successful, an HTTP status code starting with **2** and response details are displayed.
- If the request fails to be sent, an HTTP status code **4xx** or **5xx** is displayed. For details, see **Error Codes**.

Step 9 You can send more requests with different parameters and values to verify the API.

----End

Follow-Up Operations

After the API is successfully debugged, **publish** the API in a specific environment so that the API can be called by users. To ensure security, **create policies** for the API.

5.11 Authorizing API Access

APIs using app authentication can only be called by credentials that have been authorized to call them.

NOTICE

- You can authorize credentials only to call APIs that use app authentication.
- A credential can be authorized to access a maximum of 1000 APIs.

Prerequisites

- You have published an API.
- You have created an environment.
- You have created a credential.

Procedure

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 Choose **API Management** > **API Groups**.

Step 4 Click a group name.

Step 5 On the **APIs** tab, select the target API and choose **More** > **Authorize Credentials**.

Step 6 Click **Select Credentials**.

Step 7 Select an environment, search for and select desired credentials, and click **OK**. The authorized credentials are displayed on the **Authorize Credentials** page.

To cancel the authorization of a credential, click **Cancel Authorization** in the **Operation** column that contains the credential.

----End

Follow-Up Operations

After you authorize a credential for an API, the API can be called by the credential using SDKs of different programming languages.

5.12 Publishing an API

APIs can be called only after they have been published in an environment. You can publish APIs in different environments. APIG allows you to view the publication history (such as the version, description, time, and environment) of each API, and supports rollback of APIs to different historical versions.

NOTE

- If you modify a published API, you must publish it again for the modifications to take effect in the environment in which the API has been published.
- A maximum of 10 publication records of an API are retained in an environment.

Prerequisites

You have created an environment.

Publishing an API

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 Choose **API Management** > **API Groups**.

Step 4 Click a group name.

Step 5 On the **APIs** tab, select the target API and click **Publish Latest Version**.

Step 6 Select the environment where the API will be published, and enter a description.

NOTE

- If the API has already been published in the environment, publishing it again will overwrite its definition in that environment.
- If there is no environment that meets your requirements, create a new one.

Step 7 Click **OK**. After the API is published, the red exclamation mark (!) in the upper left corner of the **Publish Latest Version** button disappears.

You can remove APIs from the environments where they have been published. This operation will cause the APIs to be inaccessible in the environments. Ensure that you have notified users before this operation. To remove an API, click **Take Offline**.

----End

Viewing Publication History

Step 1 On the **APIs** tab, select the target API.

Step 2 Choose **More** > **View Publishing Records**.

Step 3 Click **View Details** in the **Operation** column of a version.

The **View Details** dialog box displays the basic information, frontend and backend request information, input and constant parameters, parameter mappings, and example responses of the API.

Step 4 To roll back the API to a historical version, click **Switch Version** in the row containing the target version, and click **Yes**.

If "current version" is displayed next to the target version, the rollback was successful.

When the API is called, configuration of the current version is used instead of the previously saved configuration.

For example, an API was published in the RELEASE environment on August 1, 2018. On August 20, 2018, the API was published in the same environment after modification. If the version published on August 1 is set as the current version, configuration of this version will be used when the API is called.

----End

FAQs About API Publishing

Do I Need to Publish an API Again After Modification?

Can I Access an API Published in a Non-RELEASE Environment?

Can I Invoke Different Backend Services by Publishing an API in Different Environments?

5.13 Taking an API Offline

You can remove APIs that you do not need from the environments where the APIs have been published.

NOTICE

This operation will cause the APIs to be inaccessible in the environments. Ensure that you have notified users before this operation.

Prerequisites

- You have created an API group and API.
- You have published the API.

Procedure

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **API Management** > **API Groups**.

Step 4 Click the name of the target API group.

- To take one API offline, select the API, and click **Take Offline** in the upper right.
- To take multiple APIs (≤ 1000) offline, click **Batch**, select the APIs, and click the Take Offline icon.

Step 5 Select the environment from which you want to take the API offline, and click **Yes**.

----End

Follow-Up Operations

After taking an API offline, delete it to release resources.

5.14 Importing and Exorting APIs

5.14.1 Restrictions and Compatibility

Note the following restrictions and compatibility issues when importing or exporting APIs on APIG:

Restrictions

- APIG parameter restrictions:
  - APIG does not support the configuration of request parameters in the **formData** and **body** locations.
  - APIG does not support the configuration of parameters **consumes** and **produces**.
  - The names of header parameters are not case-sensitive.
- Backend policy restrictions are as follows:
  - Default backend type **HTTP**: The HTTP and HTTP-VPC backends are supported.
  - Default backend type **HTTP-VPC**: The HTTP and HTTP-VPC backends are supported.
  - Default backend type **function**: Only the function backend is supported.
  - Default backend type **mock**: Only the mock backend is supported.

Compatibility

- OpenAPI is supported.

  The OpenAPI Specification (OAS) defines a standard, language-agnostic interface to RESTful APIs. OAS is formerly known as Swagger. APIG supports two OpenAPI specifications: Swagger 2.0 and OpenAPI 3.0. **For easy understanding, in the following sections, OAS refers to OpenAPI Specification (including Swagger 2.0 and OpenAPI 3.0), Swagger refers to Swagger 2.0, and OpenAPI refers to OpenAPI 3.0.**
- **Mappings** between imported or exported OAS objects and APIG's objects
- **Differences in request parameter types**
- **Differences in API request path template syntax**
- **Extended fields** supported for APIG when importing APIs

Table 5-20 Mappings between OAS objects and APIG's objects

Swagger Object	OpenAPI Object (3.0.0)	APIG Object	Import	Export
info.title	**info.title**	API group name	Importing to a new API group: a new API group name Importing to an existing API group: not used An API group name consists of 3–64 characters, starting with a letter. Only letters, digits, and underscores (_) are allowed.	API group name
info.description	**info.description**	API group description	Importing to a new API group: description about the new group Importing to an existing API group: not used	API group description
info.version	**info.version**	Version	Not used	User-defined version The current time is used as the API group name if no name is specified.
host	**server.url**	API group domain name	Not used	The first user-defined domain name of an API group is preferentially used. The independent domain name of the API group is used if the API group is not bound with any user-defined domain names.
basePath	-	-	Merged with the request path of each API	Not used

Swagger Object	OpenAPI Object (3.0.0)	APIG Object	Import	Export
paths.path	**paths.path**	API request path	Merged with **basePath** to use as an API request path	API request path
operation.operationId	**operation.operationId**	API name	API name	API name
operation.description	**operation.description**	API description	API description	API description
operation.parameters	**operation.parameters**	API frontend request parameters	API request parameters	API request parameters
operation.schemes	-	API frontend request protocol	API request protocol	API request protocol
operation.responses	**operation.responses**	-	Not used	Default response
operation.security	**operation.security**	API authentication mode	API authentication mode Used together with **x-apigateway-auth-type**	API authentication mode Used together with **x-apigateway-auth-type**

Table 5-21 Differences in request parameter types

OAS	APIG	Supported Attribute
integer long float double	number	maximum minimum default enum required description

OAS	APIG	Supported Attribute
string	string	maxLength minLength default enum required description
Other	None	None

Table 5-22 Differences in API request path template syntax

Syntax	OAS	APIG
/users/{userName}	Supported	Supported
/users/prefix-{userName} /users/{userName}-suffix /users/prefix-{userName} -suffix	Supported	Not supported for frontend request definition Supported for backend request definition
/users/{proxy+}	Not supported	Supported for frontend request definition Not supported for backend request definition

5.14.2 Importing APIs

You can import Swagger and OpenAPI APIs to a **new** or **existing** API group on APIG. Before importing APIs, complete the **extended definition** of APIG.

Precautions for Importing APIs to a New Group

When you import APIs to a new API group, the system creates an API group.

This function is suitable for importing new APIs to APIG.

Before importing APIs, ensure that the following requirements are met:

- Your API group and API quotas are sufficient.
- Use the **title** property in Swagger info and OpenAPI info to specify an API group name. The name of a new API group cannot be the same as that of an existing one.

- If a conflict exists when you import APIs, the former API is imported successfully and the latter API cannot be imported. For example, if two APIs with the same name or request path exist in the imported API definition, a success message is displayed for the first imported API, and a failure message is displayed for the API to be imported subsequently.
- If **Extended Definition Overwrite** is selected, the extended definition items (access control and request throttling policies) of an imported API will overwrite the existing extended definition items with the same name.
- Imported APIs will not be automatically published in an environment. You can choose to publish them immediately or later.

Precautions for Importing APIs to an Existing Group

When you import APIs to a specified API group, the system adds them to the API group while retaining the existing APIs.

This function is suitable for importing new or modified APIs to an existing API group.

Before importing APIs, ensure that the following requirements are met:

- Your API quota is sufficient.
- If the definition of an API you are importing is the same as that of an existing API, you can overwrite the existing API or retain it. If you leave the existing API alone, the new API will not be imported.
- If **Extended Definition Overwrite** is selected, the extended definition items (access control and request throttling policies) of an imported API will overwrite the existing extended definition items with the same name.
- Imported APIs will not be automatically published in an environment. You can choose to publish them immediately or later.

Procedure

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 Choose **API Management** > **APIs**.

Step 4 Click **Import APIs**. For details, see **Importing an API Design File**.

You can also import APIs to APIG by referring to the following examples:

- **Importing an HTTP Backend Service API**
- **Importing an HTTP VPC Backend Service API**
- **Importing a Function Backend Service API**
- **Importing a Mock Backend Service API**

----End

Importing an HTTP Backend Service API

Import the request parameter definition of an HTTP backend service API that uses the GET method and is accessed through IAM authentication.

Swagger example:

```
swagger: "2.0"
info:
  title: "importHttpEndpoint10"
  description: "import apis"
  version: "1.0"
host: "api.account.com"
paths:
  '/http/{userId}':
    get:
      operationId: "getUser3"
      description: "get user by userId"
      security:
      - apig-auth-iam: []
      schemes:
      - https
      parameters:
      - name: "test"
        description: "authorization token"
        type: "string"
        in: "header"
        required: true
      - name: "userId"
        description: "user id"
        type: "string"
        in: "path"
        required: true
      responses:
        "200":
          description: "user information"
      x-apigateway-request-type: "public"
      x-apigateway-cors: true
      x-apigateway-is-send-fg-body-base64: true
      x-apigateway-match-mode: "NORMAL"
      x-apigateway-backend:
        type: "HTTP"
        parameters:
        - name: "userId"
          value: "userId"
          in: "query"
          origin: "REQUEST"
          description: "user id"
        - name: "X-Invoke-User"
          value: "apigateway"
          in: "header"
          origin: "CONSTANT"
          description: "invoke user"
        httpEndpoints:
          address: "example.com"
          scheme: "http"
          method: "GET"
          path: "/users"
          timeout: 30000
securityDefinitions:
  apig-auth-app:
    in: header
    name: Authorization
    type: apiKey
    x-apigateway-auth-type: AppSigv1
  apig-auth-iam:
    in: header
    name: unused
    type: apiKey
    x-apigateway-auth-type: IAM
```

OpenAPI example:

```
openapi: 3.0.0
info:
  title: importHttpEndpoint10
  version: '1.0'
servers:
  - url: >-
      http://abc.com
  - url: >-
      https://abc.com
paths:
  '/http/{userId}':
    get:
      description: get user by userId
      operationId: getUser3
      parameters:
        - description: authorization token
          example: ''
          in: header
          name: test
          required: true
          schema:
            maxLength: 0
            maximum: 0
            minimum: 0
            type: string
          x-apigateway-pass-through: always
        - description: user id
          example: ''
          in: path
          name: userId
          required: true
          schema:
            maxLength: 0
            maximum: 0
            minimum: 0
            type: string
          x-apigateway-pass-through: always
      responses:
        default-cors:
          description: response example
          x-apigateway-result-failure-sample: ''
          x-apigateway-result-normal-sample: ''
      security:
        - apig-auth-iam: []
      servers:
        - url: >-
            https://abc.com
      x-apigateway-backend:
        httpEndpoints:
          address: example.com
          description: ''
          enableClientSsl: false
          method: GET
          path: /users
          retryCount: '-1'
          scheme: http
          timeout: 30000
        parameters:
          - description: invoke user
            in: HEADER
            name: X-Invoke-User
            origin: CONSTANT
            value: apigateway
          - description: user id
            in: QUERY
            name: userId
            origin: REQUEST
            value: userId
        type: HTTP
      x-apigateway-cors: true
      x-apigateway-is-send-fg-body-base64: true
      x-apigateway-match-mode: NORMAL
      x-apigateway-request-type: public
      x-apigateway-response: default
components:
  responses:
    default-cors:
      description: response example
      headers:
        Access-Control-Allow-Origin:
          schema:
            default: '*'
            type: string
  securitySchemes:
    apig-auth-app:
      in: header
      name: Authorization
      type: apiKey
      x-apigateway-auth-type: AppSigv1
    apig-auth-app-header:
      in: header
      name: Authorization
      type: apiKey
      x-apigateway-auth-opt:
        appcode-auth-type: header
      x-apigateway-auth-type: AppSigv1
    apig-auth-iam:
      in: header
      name: unused
      type: apiKey
      x-apigateway-auth-type: IAM
  x-apigateway-responses:
    default: {}
```

Importing an HTTP VPC Backend Service API

Import the request parameter definition of an HTTP VPC backend service API that uses the ANY method and is accessed through app authentication.

Swagger example:

```
swagger: "2.0"
info:
  title: "importHttpVpcEndpoint"
  description: "import apis"
  version: "1.0"
host: "api.account.com"
paths:
  '/http-vpc':
    x-apigateway-any-method:
      operationId: "userOperation"
      description: "user operation resource"
      security:
      - apig-auth-app: []
      schemes:
      - https
      parameters:
      - name: "Authorization"
        description: "authorization signature"
        type: "string"
        in: "header"
        required: true
      responses:
        "default":
          description: "endpoint response"
      x-apigateway-request-type: "public"
      x-apigateway-cors: true
      x-apigateway-is-send-fg-body-base64: true
      x-apigateway-match-mode: "SWA"
      x-apigateway-backend:
        type: "HTTP-VPC"
        parameters:
        - name: "X-Invoke-User"
          value: "apigateway"
          in: "header"
          origin: "CONSTANT"
          description: "invoke user"
        httpVpcEndpoints:
          name: "userVpc"
          scheme: "http"
          method: "GET"
          path: "/users"
          timeout: 30000
securityDefinitions:
  apig-auth-app:
    in: header
    name: Authorization
    type: apiKey
    x-apigateway-auth-type: AppSigv1
  apig-auth-iam:
    in: header
    name: unused
    type: apiKey
    x-apigateway-auth-type: IAM
```

OpenAPI example:

```
openapi: 3.0.0
info:
  description: import apis
  title: importHttpVpcEndpoint
  version: '1.0'
servers:
  - url: >-
      http://abc.com
  - url: >-
      https://abc.com
paths:
  /http-vpc:
    x-apigateway-any-method:
      description: user operation resource
      operationId: userOperation
      parameters:
        - description: authorization signature
          example: ''
          in: header
          name: Authorization
          required: true
          schema:
            maxLength: 0
            maximum: 0
            minimum: 0
            type: string
          x-apigateway-pass-through: always
      responses:
        default-cors:
          description: response example
          x-apigateway-result-failure-sample: ''
          x-apigateway-result-normal-sample: ''
      security:
        - apig-auth-app: []
      servers:
        - url: >-
            https://abc.com
      x-apigateway-backend:
        httpVpcEndpoints:
          cascade_flag: false
          description: ''
          enableClientSsl: false
          method: GET
          name: userVpc
          path: /users
          retryCount: '-1'
          scheme: http
          timeout: 30000
        parameters:
          - description: invoke user
            in: HEADER
            name: X-Invoke-User
            origin: CONSTANT
            value: apigateway
        type: HTTP-VPC
      x-apigateway-cors: true
      x-apigateway-is-send-fg-body-base64: true
      x-apigateway-match-mode: SWA
      x-apigateway-request-type: public
components:
  responses:
    default-cors:
      description: response example
      headers:
        Access-Control-Allow-Origin:
          schema:
            default: '*'
            type: string
  securitySchemes:
    apig-auth-app:
      in: header
      name: Authorization
      type: apiKey
      x-apigateway-auth-type: AppSigv1
    apig-auth-app-header:
      in: header
      name: Authorization
      type: apiKey
      x-apigateway-auth-opt:
        appcode-auth-type: header
      x-apigateway-auth-type: AppSigv1
    apig-auth-iam:
      in: header
      name: unused
      type: apiKey
      x-apigateway-auth-type: IAM
  x-apigateway-responses: {}
```

Importing a Function Backend Service API

Import the request parameter definition of a FunctionGraph backend service API that uses the GET method and is accessed through IAM authentication.

Swagger example:

```
swagger: "2.0"
info:
  title: "importFunctionEndpoint"
  description: "import apis"
  version: "1.0"
host: "api.account.com"
paths:
  '/function/{name}':
    get:
      operationId: "invokeFunction"
      description: "invoke function by name"
      security:
      - apig-auth-iam: []
      schemes:
      - https
      parameters:
      - name: "test"
        description: "authorization token"
        type: "string"
        in: "header"
        required: true
      - name: "name"
        description: "function name"
        type: "string"
        in: "path"
        required: true
      responses:
        "200":
          description: "function result"
      x-apigateway-request-type: "public"
      x-apigateway-cors: true
      x-apigateway-is-send-fg-body-base64: true
      x-apigateway-match-mode: "NORMAL"
      x-apigateway-backend:
        type: "FUNCTION"
        parameters:
        - name: "functionName"
          value: "name"
          in: "query"
          origin: "REQUEST"
          description: "function name"
        - name: "X-Invoke-User"
          value: "apigateway"
          in: "header"
          origin: "CONSTANT"
          description: "invoke user"
        functionEndpoints:
          function-urn: "your function urn address"
          version: "your function version"
          invocation-type: "async"
          timeout: 30000
securityDefinitions:
  apig-auth-app:
    in: header
    name: Authorization
    type: apiKey
    x-apigateway-auth-type: AppSigv1
  apig-auth-iam:
    in: header
    name: unused
    type: apiKey
    x-apigateway-auth-type: IAM
```

OpenAPI example:

```
openapi: 3.0.0
info:
  description: import apis
  title: importHttpEndpoint
  version: '1.0'
servers:
  - url: >-
      http://api.account.com
  - url: >-
      https://api.account.com
paths:
  /function/{name}:
    get:
      description: invoke function by name
      operationId: invokeFunction
      parameters:
        - description: function name
          in: path
          name: name
          required: true
          schema:
            maxLength: 0
            maximum: 0
            minimum: 0
            type: string
          x-apigateway-pass-through: always
          example: ''
        - description: authorization token
          in: header
          name: test
          required: true
          schema:
            maxLength: 0
            maximum: 0
            minimum: 0
            type: string
          x-apigateway-pass-through: always
          example: ''
      responses:
        default-cors:
          description: response example
          x-apigateway-result-failure-sample: ''
          x-apigateway-result-normal-sample: ''
      security:
        - apig-auth-iam: []
      servers:
        - url: >-
            https://api.account.com
      x-apigateway-backend:
        functionEndpoints:
          alias-urn: ''
          description: ''
          function-urn: "your function urn address"
          invocation-type: async
          network-type: V1
          timeout: 30000
          version: "your function version"
        parameters:
          - description: invoke user
            in: HEADER
            name: X-Invoke-User
            origin: CONSTANT
            value: apigateway
          - description: function name
            in: QUERY
            name: functionName
            origin: REQUEST
            value: name
        type: FUNCTION
      x-apigateway-cors: true
      x-apigateway-is-send-fg-body-base64: true
      x-apigateway-match-mode: NORMAL
      x-apigateway-request-type: public
      x-apigateway-response: default
components:
  responses:
    default-cors:
      description: response example
      headers:
        Access-Control-Allow-Origin:
          schema:
            default: '*'
            type: string
  securitySchemes:
    apig-auth-app:
      in: header
      name: Authorization
      type: apiKey
      x-apigateway-auth-type: AppSigv1
    apig-auth-iam:
      in: header
      name: unused
      type: apiKey
      x-apigateway-auth-type: IAM
  x-apigateway-responses:
    default: {}
```

Importing a Mock Backend Service API

Import the definition of a Mock backend service API that uses the GET method and is accessed without authentication.

Swagger example:

```
swagger: "2.0"
info:
  title: "importMockEndpoint"
  description: "import apis"
  version: "1.0"
host: "api.account.com"
paths:
  '/mock':
    get:
      operationId: "mock"
      description: "mock test"
      schemes:
      - http
      responses:
        "200":
          description: "mock result"
      x-apigateway-request-type: "private"
      x-apigateway-cors: true
      x-apigateway-is-send-fg-body-base64: true
      x-apigateway-match-mode: "NORMAL"
      x-apigateway-backend:
        type: "MOCK"
        mockEndpoints:
          result-content: "{\"message\": \"mocked\"}"
securityDefinitions:
  apig-auth-app:
    in: header
    name: Authorization
    type: apiKey
    x-apigateway-auth-type: AppSigv1
  apig-auth-iam:
    in: header
    name: unused
    type: apiKey
    x-apigateway-auth-type: IAM
```

OpenAPI example:

```
openapi: 3.0.0
info:
  description: import apis
  title: importHttpVpcEndpoint
  version: '1.0'
servers:
  - url: >-
      http://abc.com
  - url: >-
      https://abc.com
paths:
  /mock:
    get:
      description: mock test
      operationId: mock
      responses:
        default-cors:
          description: response example
          x-apigateway-result-failure-sample: ''
          x-apigateway-result-normal-sample: ''
      servers:
        - url: >-
            http://abc.com
      x-apigateway-backend:
        mockEndpoints:
          description: ''
          result-content: '{"message": "mocked"}'
        type: MOCK
      x-apigateway-cors: true
      x-apigateway-is-send-fg-body-base64: true
      x-apigateway-match-mode: NORMAL
      x-apigateway-request-type: private
      x-apigateway-response: default
components:
  responses:
    default-cors:
      description: response example
      headers:
        Access-Control-Allow-Origin:
          schema:
            default: '*'
            type: string
  securitySchemes:
    apig-auth-app:
      in: header
      name: Authorization
      type: apiKey
      x-apigateway-auth-type: AppSigv1
    apig-auth-app-header:
      in: header
      name: Authorization
      type: apiKey
      x-apigateway-auth-opt:
        appcode-auth-type: header
      x-apigateway-auth-type: AppSigv1
    apig-auth-iam:
      in: header
      name: unused
      type: apiKey
      x-apigateway-auth-type: IAM
  x-apigateway-responses:
    default: {}
```

Follow-Up Operations

Publish the imported APIs in an environment so that they can be called by users.

5.14.3 Exporting APIs

You can export APIs one by one or in batches as JSON, YAML, or YML files.

Procedure

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **API Management** > **API Groups**. Click a group name and click **Export**.

Or choose **API Management** > **APIs**, and click **Export APIs**.

Step 4 Set the export parameters.

Table 5-23 Parameters for exporting APIs

Parameter	Description
API Group	Select the group of which APIs will be exported.
Environment	Select the environment where the APIs to be exported have been published.
APIs	By default, all APIs in the group that have been published in the selected environment are exported. To export only specific APIs, click **Select APIs**, and specify the APIs you want to export.
API Definition	● **Basic**: The basic definition of an API is composed of the request and response definitions. It does not include the backend definition. The request definition includes both standard and extended Swagger fields. This function can generate a Swagger or OpenAPI API definition file. ● **Full**: The full definition of an API is composed of the request, backend, and response definitions. This function can be used to back up the full definition of an API as a Swagger or OpenAPI file. ● **Extended**: The extended definition of an API is composed of the request, backend, and response definitions as well as the request throttling policy, access control policy, and other configurations of the API.
Format	Select **JSON**, **YAML**, or **YML**.
Version	Set the version of the APIs to be exported. If you do not specify a version, the version will be set as the current date and time.
OpenAPI Version	Export Swagger 2.0 or OpenAPI 3.0 APIs.

Step 5 Click **Export**. The export result is displayed on the right of the page and the API file is automatically downloaded.

----End

5.14.4 Extended Definition

5.14.4.1 x-apigateway-auth-type

Meaning: Swagger-based apiKey authentication format, which defines an authentication mode provided by APIG.

Scope of effect: **Security Scheme Object (2.0)**/**Security Scheme Object (3.0)**

Swagger:

```
securityDefinitions:
  apig-auth-app:
    in: header
    name: Authorization
    type: apiKey
    x-apigateway-auth-type: AppSigv1
  apig-auth-iam:
    in: header
    name: unused
    type: apiKey
    x-apigateway-auth-type: IAM
```

OpenAPI example:

```
securitySchemes:
    apig-auth-app:
      in: header
      name: Authorization
      type: apiKey
      x-apigateway-auth-type: AppSigv1
    apig-auth-iam:
      in: header
      name: unused
      type: apiKey
      x-apigateway-auth-type: IAM
```

Table 5-24 Parameter description

Parameter	Mandatory	Type	Description
x-apigateway-auth-type	Yes	String	Authentication mode used on APIG. **AppSigv1** and **IAM** are supported.
type	Yes	String	Authentication type. Only **apiKey** is supported.
name	Yes	String	Name of the parameter for authentication.
in	Yes	String	Only **header** is supported.
description	No	String	Description about the authentication.

5.14.4.2 x-apigateway-request-type

Meaning: API request type, which can be **public** or **private**.

Scope of effect: Operation Object (2.0)/Operation Object (3.0)

Example:

```
paths:
  '/path':
    get:
      x-apigateway-request-type: 'public'
```

Table 5-25 Parameter description

Parameter	Mandatory	Type	Description
x-apigateway-request-type	Yes	String	API visibility. The options include **public** and **private**. ● **public**: The API can be made available for sale. ● **private**: The API will not be available for sale.

5.14.4.3 x-apigateway-match-mode

Meaning: Request URL matching mode, which can be **NORMAL** or **SWA**.

Scope of effect: **Operation Object (2.0)**/**Operation Object (3.0)**

Example:

```
paths:
  '/path':
    get:
      x-apigateway-match-mode: 'SWA'
```

Table 5-26 Parameter description

Parameter	Mandatory	Type	Description
x-apigateway-match-mode	Yes	String	API matching mode. The options include **SWA** and **NORMAL**. ● **SWA**: prefix match. For example, both **/prefix/foo** and **/prefix/bar** match **/prefix**, but **/prefixpart** does not match. ● **NORMAL**: exact match.

5.14.4.4 x-apigateway-cors

Meaning: Specifies whether CORS is supported. The value is of the Boolean type.

Scope of effect: **Operation Object (2.0)**/**Operation Object (3.0)**

Example:

```
paths:
  '/path':
    get:
      x-apigateway-cors: true
```

Table 5-27 Parameter description

Parameter	Mandatory	Type	Description
x-apigateway-cors	Yes	boolean	Whether to support CORS. ● **true**: support ● **false**: not support

For the API request for enabling CORS, the headers listed in the following table will be added to the response.

Header	Value	Description
Access-Control-Max-Age	172800	Maximum time the response of a preflight request can be cached. Unit: s
Access-Control-Allow-Origin	*	Requests from any domain are allowed.
Access-Control-Allow-Headers	X-Sdk-Date, X-Sdk-Nonce, X-Proxy-Signed-Headers, X-Sdk-Content-Sha256, X-Forwarded-For, Authorization, Content-Type, Accept, Accept-Ranges, Cache-Control, and Range	Headers that can be used by a formal request.
Access-Control-Allow-Methods	GET, POST, PUT, DELETE, HEAD, OPTIONS, and PATCH	Methods that can be used by a formal request.

5.14.4.5 x-apigateway-is-send-fg-body-base64

Meaning: Whether to perform Base64 encoding on the request body used for interaction with FunctionGraph. The value is of the Boolean type.

Scope of effect: **Operation Object (2.0)**/**Operation Object (3.0)**

Example:

```
paths:
  '/path':
    get:
      "x-apigateway-is-send-fg-body-base64": true
```

Table 5-28 Parameter description

Parameter	Mandatory	Type	Description
x-apigateway-is-send-fg-body-base64	No	boolean	Specifies whether to perform Base64 encoding on the request body for interaction with FunctionGraph. ● **true**: yes ● **false**: no

5.14.4.6 x-apigateway-any-method

Meaning: API request method used by default if no HTTP request method is specified.

Scope of effect: Path Item Object (2.0)/Path Item Object (3.0)

Example:

```
paths:
  '/path':
    get:
      produces:
        - application/json
      responses:
        "200":
          description: "get response"
    x-apigateway-any-method:
      produces:
        - application/json
      responses:
        "200":
          description: "any response"
```

Table 5-29 Parameter description

Parameter	Mandatory	Type	Description
x-apigateway-any-method	No	String	Request method.

5.14.4.7 x-apigateway-backend

Meaning: API backend definition.

Scope of effect: Operation Object (2.0)/Operation Object (3.0)

Example:

```
paths:
  '/users/{userId}':
    get:
```

```
   produces:
     - "application/json"
   responses:
     default:
       description: "default response"
   x-apigateway-request-type: "public"
   x-apigateway-backend:
     type: "backend endpoint type"
```

Table 5-30 Parameter description

Parameter	Mandatory	Type	Description
x-apigateway-backend	Yes	String	Backend service definition.
type	Yes	String	Backend service type. The options include **HTTP**, **HTTP-VPC**, **FUNCTION**, and **MOCK**.
parameters	No	**x-apigateway-backend.parameters**	Backend parameters.
httpEndpoints	No	**x-apigateway-backend.httpEndpoints**	HTTP backend service definition.
httpVpcEndpoints	No	**x-apigateway-backend.httpVpcEndpoints**	HTTP VPC backend service definition.
functionEndpoints	No	**x-apigateway-backend.functionEndpoints**	Function backend service definition.
mockEndpoints	No	**x-apigateway-backend.mockEndpoints**	Mock backend service definition.

5.14.4.8 x-apigateway-backend.parameters

Meaning: API backend service definition.

Scope of effect: **x-apigateway-backend**

Example:

```
paths:
  '/users/{userId}':
```

```
  get:
    produces:
      - "application/json"
    parameters:
      - name: "X-Auth-Token"
        description: "Authentication token"
        type: "string"
        in: "header"
        required: true
      - name: "userId"
        description: "Username"
        type: "string"
        in: "path"
        required: true
    responses:
      default:
        description: "default response"
    x-apigateway-request-type: "public"
    x-apigateway-backend:
      type: "HTTP"
      parameters:
        - name: "userId"
          value: "userId"
          in: "query"
          origin: "REQUEST"
        description: "Username"
        - name: "X-Invoke-User"
          value: "apigateway"
          in: "header"
          origin: "CONSTANT"
        description: "Caller"
```

Table 5-31 Parameter description

Parameter	Mandatory	Type	Description
name	Yes	String	Parameter name, which consists of a maximum of 32 bytes, starting with a letter. Only letters, digits, periods (.), hyphens (-), and underscores (_) are allowed. The names of header parameters are not case-sensitive.
value	Yes	String	Parameter value, which is a parameter name if the parameter comes from a request.
in	Yes	String	Parameter location, which can be **header**, **query**, or **path**.
origin	Yes	String	Parameter mapping source. The options include **REQUEST** and **CONSTANT**.
description	No	String	Parameter meaning.

5.14.4.9 x-apigateway-backend.httpEndpoints

Meaning: HTTP backend service definition.

Scope of effect: **x-apigateway-backend**

Example:

```
paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      parameters:
        - name: "X-Auth-Token"
          description: "Authentication token"
          type: "string"
          in: "header"
          required: true
      responses:
        default:
          description: "default response"
      x-apigateway-request-type: "public"
      x-apigateway-backend:
        type: "HTTP"
        httpEndpoints:
          address: "example.com"
          scheme: "http"
          method: "GET"
          path: "/users"
          timeout: 30000
```

Table 5-32 Parameter description

Parameter	Mandatory	Type	Description
address	Yes	Array	Backend service address. The format is *<Domain name or IP address>:[Port number]*
scheme	Yes	String	Backend request protocol. HTTP and HTTPS are supported.
method	Yes	String	Backend request method. The options include **GET**, **POST**, **PUT**, **DELETE**, **HEAD**, **OPTIONS**, **PATCH**, and **ANY**.
path	Yes	String	Backend request path, which can contain variables.
timeout	No	Number	Backend request timeout in milliseconds. The range is 1–60,000, and the default value is **5000**.

5.14.4.10 x-apigateway-backend.httpVpcEndpoints

Meaning: HTTP VPC backend service definition.

Scope of effect: x-apigateway-backend

Example:

```
paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      parameters:
        - name: "X-Auth-Token"
          description: "Authentication token"
          type: "string"
          in: "header"
          required: true
      responses:
        default:
          description: "default response"
      x-apigateway-request-type: "public"
      x-apigateway-backend:
        type: "HTTP-VPC"
        httpVpcEndpoints:
          name: "vpc-test-1"
          scheme: "http"
          method: "GET"
          path: "/users"
          timeout: 30000
```

Table 5-33 Parameter description

Parameter	Mandatory	Type	Description
name	Yes	Array	VPC channel name.
scheme	Yes	String	Backend request protocol. HTTP and HTTPS are supported.
method	Yes	String	Backend request method. The options include **GET**, **POST**, **PUT**, **DELETE**, **HEAD**, **OPTIONS**, **PATCH**, and **ANY**.
path	Yes	String	Backend request path, which can contain variables.
timeout	No	Number	Backend request timeout in milliseconds. The range is 1–60,000, and the default value is **5000**.

5.14.4.11 x-apigateway-backend.functionEndpoints

Meaning: Function backend service definition.

Scope of effect: x-apigateway-backend

Example:

```
paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      parameters:
        - name: "X-Auth-Token"
          description: "Authentication token"
          type: "string"
          in: "header"
          required: true
      responses:
        default:
          description: "default response"
      x-apigateway-request-type: "public"
      x-apigateway-backend:
        type: "FUNCTION"
        functionEndpoints:
          version: "v1"
          function-urn: ""
          invocation-type: "synchronous"
          timeout: 30000
```

Table 5-34 Parameter description

Parameter	Mandatory	Type	Description
function-urn	Yes	String	Function URN.
version	Yes	String	Function version.
invocation-type	Yes	String	Function invocation type. The value can be **async** or **sync**.
timeout	No	Number	Function timeout in milliseconds. The range is 1–60,000, and the default value is **5000**.

5.14.4.12 x-apigateway-backend.mockEndpoints

Meaning: Mock backend service definition.

Scope of effect: **x-apigateway-backend**

Example:

```
paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      parameters:
        - name: "X-Auth-Token"
          description: "Authentication token"
          type: "string"
          in: "header"
          required: true
      responses:
        default:
          description: "default response"
      x-apigateway-request-type: "public"
      x-apigateway-backend:
        type: "MOCK"
        mockEndpoints:
          result-content: "mocked"
```

Table 5-35 Parameter description

Parameter	Mandatory	Type	Description
result-content	Yes	String	Mock response.

5.14.4.13 x-apigateway-backend-policies

Meaning: API backend policy.

Scope of effect: Operation Object (2.0)/Operation Object (3.0)

Example:

```
paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      responses:
        default:
          description: "default response"
      x-apigateway-request-type: "public"
      x-apigateway-backend:
        type: "backend endpoint type"
      x-apigateway-backend-policies:
        - type: "backend endpoint type"
          name: "backend policy name"
          conditions:
            - type: "equal/enum/pattern",
              value: "string",
              origin: "source/request_parameter",
              parameter_name: "string"
```

Table 5-36 Parameter description

Parameter	Mandatory	Type	Description
x-apigateway-backend-policies	No	x-apigateway-backend-policies	Backend policies.
type	Yes	String	Backend service type. The options include **HTTP**, **HTTP-VPC**, **FUNCTION**, and **MOCK**.
name	Yes	String	Backend policy name.

Parameter	Mandatory	Type	Description
parameters	No	**x-apigateway-backend.parameters**	Backend parameters.
httpEndpoints	No	**x-apigateway-backend.httpEndpoints**	HTTP service definition.
httpVpcEndpoints	No	**x-apigateway-backend.httpVpcEndpoints**	HTTP-VPC service definition.
functionEndpoints	No	**x-apigateway-backend.functionEndpoints**	Function service definition.
mockEndpoints	No	**x-apigateway-backend.mockEndpoints**	Mock service definition.
conditions	Yes	**x-apigateway-backend-policies.conditions**	Policy condition array.

5.14.4.14 x-apigateway-backend-policies.conditions

Meaning: API backend policy conditions.

Scope of effect: **x-apigateway-backend-policies**

Example:

```
paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      responses:
        default:
          description: "default response"
      x-apigateway-request-type: "public"
      x-apigateway-backend:
        type: "backend endpoint type"
      x-apigateway-backend-policies:
        - type: "backend endpoint type"
```

```
     name: "backend policy name"
     conditions:
       - type: "equal/enum/pattern",
         value: "string",
         origin: "source/request_parameter",
         parameter_name: "string"
```

Table 5-37 Parameter description

Parameter	Mandatory	Type	Description
type	Yes	String	Policy condition type. The options include **equal**, **enum**, and **pattern**.
value	Yes	String	Policy condition value.
origin	Yes	String	Policy condition source. The options include **source** and **request**.
parameter	No	String	Input parameter name if the **origin** parameter is set to **request**.

5.14.4.15 x-apigateway-ratelimit

Meaning: Request throttling policy.

Scope of effect: **Operation Object (2.0)**/**Operation Object (3.0)**

Example:

```
paths:
  '/path':
    get:
      x-apigateway-ratelimit: 'customRatelimitName'
```

Table 5-38 Parameter description

Parameter	Mandatory	Type	Description
x-apigateway-ratelimit	No	String	Request throttling policy.

5.14.4.16 x-apigateway-ratelimits

Meaning: Mapping between a request throttling policy name and limit values.

Scope of effect: **Swagger Object**

Example:

```
x-apigateway-ratelimits:
  customRatelimitName:
    api-limit: 200
    app-limit: 200
    user-limit: 200
    ip-limit: 200
    interval: 1
    unit: second/minute/hour
    shared: true
    special:
      - type: APP
        limit: 100
        instance: xxxxxxxxx
```

Table 5-39 Parameter description

Parameter	Mandatory	Type	Description
customRatelimitName	No	**x-apigateway-ratelimits.policy**	Name of a request throttling policy. To use the policy, set **x-apigateway-ratelimit** to the policy name.

5.14.4.17 x-apigateway-ratelimits.policy

Meaning: Definition of a request throttling policy.

Scope of effect: **x-apigateway-ratelimits**

Example:

```
x-apigateway-ratelimits:
  customRatelimitName:
    api-limit: 200
    app-limit: 200
    user-limit: 200
    ip-limit: 200
    interval: 1
    unit: MINUTE
    shared: false
    special:
      - type: USER
        limit: 100
        instance: xxxxxxx
```

Table 5-40 Parameter description

Parameter	Mandatory	Type	Description
api-limit	Yes	Number	Maximum number of times an API can be called.
user-limit	No	Number	Maximum number of times the API can be called by a user.
app-limit	No	Number	Maximum number of times the API can be called by an app.

Parameter	Mandatory	Type	Description
ip-limit	No	Number	Maximum number of times the API can be called by an IP address.
interval	Yes	Number	Throttling period.
unit	Yes	String	Throttling unit, which can be **SECOND**, **MINUTE**, **HOUR**, or **DAY**.
shared	No	Boolean	Whether to share the throttling limits among APIs.
special	No	**x-apigateway-ratelimits.policy.special** Array	Special request throttling policy.

5.14.4.18 x-apigateway-ratelimits.policy.special

Meaning: Definition of a special request throttling policy.

Scope of effect: **x-apigateway-ratelimits.policy**

Example:

```
x-apigateway-ratelimits:
  customRatelimitName:
    api-limit: 200
    app-limit: 200
    user-limit: 200
    ip-limit: 200
    interval: 1
    unit: MINUTE
    shared: false
    special:
      - type: USER
        limit: 100
        instance: xxxxxxxx
```

Table 5-41 Parameter description

Parameter	Mandatory	Type	Description
type	Yes	String	Special request throttling policy type, which can be **APP** or **USER**.
limit	Yes	Number	Access limit.
instance	Yes	String	ID of an excluded app or user.

5.14.4.19 x-apigateway-access-control

Meaning: Access control policy.

Scope of effect: **Operation Object (2.0)**/**Operation Object (3.0)**

Example:

```
paths:
  '/path':
    get:
      x-apigateway-access-control: 'customAccessControlName'
```

Table 5-42 Parameter description

Parameter	Mandatory	Type	Description
x-apigateway-access-control	No	String	Access control policy.

5.14.4.20 x-apigateway-access-controls

Meaning: Mapping between an access control policy name and limit settings.

Scope of effect: **Swagger Object**

Example:

```
x-apigateway-access-controls:
  customAccessControlName:
    acl-type: "DENY"
    entity-type: "IP"
    value: 127.0.0.1,192.168.0.1/16
```

Table 5-43 Parameter description

Parameter	Mandatory	Type	Description
customAccessControlName	No	**x-apigateway-access-controls.policy**	Name of an access control policy. To use the policy, set **x-apigateway-access-control** to the policy name.

5.14.4.21 x-apigateway-access-controls.policy

Meaning: Definition of an access control policy.

Scope of effect: **x-apigateway-access-controls**

Example:

```
x-apigateway-access-controls:
  customAccessControlName:
    acl-type: "DENY"
    entity-type: "IP"
    value: 127.0.0.1,192.168.0.1/16
```

Table 5-44 Parameter description

Parameter	Mandatory	Type	Description
acl-type	Yes	String	Access control effect. The options include **PERMIT** and **DENY**.
entity-type	Yes	String	Access control object. Only IP addresses are supported.
value	Yes	String	Access control values, which are separated with commas (,).

5.14.4.22 x-apigateway-plugins

Meaning: API plug-in service.

Scope of effect: Operation Object (2.0)/Operation Object (3.0)

Example:

```
paths:
  '/path':
    get:
      x-apigateway-plugins: ['Plugin_mock']
x-apigateway-plugins
```

Table 5-45 Parameter description

Parameter	Mandatory	Type	Description
x-apigateway-plugins	No	Array	List of plug-ins bound to the API.

5.15 Viewing APIs

The **APIs** page displays all APIs of the current gateway, including the URL, running environment, and authentication mode.

Procedure

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 Modify, publish, and debug APIs of the gateway.

Step 4 In the navigation pane, choose **API Management** > **APIs**.

Step 5 Click an API name to go to the details page of the group to which the API belongs. For details about how to create an API, manage domain names, and set environment variables, see the preceding sections.

----End

5.16 HTTP 2.0

APIG supports HTTP/2, which is a major revision of HTTP and was originally named HTTP 2.0. It provides binary encoding, request multiplexing over a single connection, and request header compression, improving transmission performance and throughput with a lower latency.

NOTE

- HTTP 2.0 strongly depends on network stability. To use HTTP 2.0, ensure that your network is stable and your client supports this protocol.
- If your gateway does not support HTTP 2.0, contact technical support to upgrade it.
- To disable HTTP 2.0, turn off **HTTP/2** under the **request_custom_config** parameter on the **Parameters** tab page of the APIG console.

- Binary encoding

  Unlike HTTP 1.x where data is transmitted in text format, data in HTTP 2.0 is split into messages and frames for binary encoding. Compared with string (text) parsing, binary parsing is easier and less error-prone and delivers higher transmission performance.
- Multiplexing

  With binary encoding, HTTP 2.0 no longer relies on multiple connections to process and send requests and responses concurrently.

  For the same domain name, all requests are completed on a single connection, and each connection can process any number of messages. A message consists of one or more frames, which can be sent out of order and finally recombined based on the stream ID in the header of each frame. This shortens the latency and improves the efficiency.
- Header compression

  HTTP 2.0 uses an encoder to reduce the size of the headers to transmit. Both the client and server store a header field table to avoid transmitting same headers repeatedly, achieving high throughput.

6 API Policies

6.1 Creating a Policy and Binding It to APIs

APIG provides flexible API control policies.

NOTICE

Policy parameters will be stored as plaintext. To prevent information leakage, do not contain sensitive information in these parameters.

Guidelines

- An API can be bound with only one policy of the same type.
- Policies are independent of APIs. A policy takes effect for an API only after they are bound to each other. When binding a policy to an API, you must specify an environment where the API has been published. The policy takes effect for the API only in the specified environment.
- After you bind a policy to an API, unbind the policy from the API, or update the policy, you do not need to publish the API again.
- Taking an API offline does not affect the policies bound to it. The policies are still bound to the API if the API is published again.
- Policies that have been bound to APIs cannot be deleted.

Creating a Policy

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **API Management** > **API Policies**.

Step 4 On the **Policies** tab, click **Create Policy**.

Step 5 Click the desired policy type.

- **Plug-in policies**

  Set the policy information.

Table 6-1 Policy configuration

Parameter	Description
Name	Enter a policy name that conforms to specific rules to facilitate search.
Type	Type of the policy, which determines the extension capabilities. **NOTE** If a policy type is not supported by your gateway, contact technical support to upgrade the gateway to the latest version. – **CORS**: Provides the capabilities of specifying preflight request headers and response headers and automatically creating preflight request APIs for cross-origin API access. – **HTTP Response Header Management**: Enables you to customize HTTP response headers that will be displayed in an API response. – **Request Throttling 2.0**: Limits the number of times that an API can be called within a specific time period. Parameter-based, basic, and excluded throttling is supported. – **Kafka Log Push**: Pushes API calling logs to Kafka so that you can view these logs. – **Circuit Breaker**: Protects your backend service when a performance issue occurs. – **Third-Party Authorizer**: Authenticates API requests with your own service.
Description	Description about the plug-in.
Policy Content	Content of the plug-in, which can be configured in a form or using a script. The plug-in content varies depending on the plug-in type: – **CORS** – **HTTP Response Header Management** – **Request Throttling 2.0** – **Kafka Log Push** – **Circuit Breaker** – **Third-Party Authorizer**

- **Traditional policies**

  The policy content varies depending on the policy type:

– **Request Throttling**
– **Access Control**
– **Signature Keys**

Step 6 Click **OK**.

- To clone this policy, click **Clone** in the **Operation** column.

  NOTE

  – The name of a cloned policy cannot be the same as that of any existing policy.
  – **Request throttling** and **signature key** policies cannot be cloned.

- After the policy is created, perform the operations described in **Binding the Policy to APIs** for the policy to take effect for the API.

----End

Binding the Policy to APIs

Step 1 Click a policy name to go to the policy details page.

Step 2 In the **APIs** area, select an environment and click **Select APIs**.

Step 3 Select the API group, environment, and required APIs.

Step 4 Click **OK**.

- If an API no longer needs this policy, click **Unbind** in the row that contains the API.
- If there are multiple APIs that no longer need this policy, select these APIs, and click **Unbind** above the API list. You can unbind a policy from a maximum of 1000 APIs at a time.

----End

6.2 CORS

For security purposes, the browser restricts cross-domain requests from being initiated from a page script. In this case, the page can access only the resources from the current domain. CORS allows the browser to send XMLHttpRequest to the server in a different domain. For details about CORS, see **CORS**.

The CORS plug-in provides the capabilities of specifying preflight request headers and response headers and automatically creating preflight request APIs for cross-origin API access.

NOTE

If your gateway does not support this policy, contact technical support to upgrade the gateway to the latest version.

Usage Guidelines

- You have understood the **Guidelines for Using Plug-ins**.
- APIs with the same request path in an API group can only be bound with the same CORS plug-in policy.

- If you have enabled CORS for an API and have also bound the CORS plug-in to the API, the CORS plug-in will be used.
- You cannot bind the CORS plug-in to APIs with the same request path as another API that uses the OPTIONS method.
- When you bind a plug-in policy to an API (see **Binding the Policy to APIs**), ensure that the request method of the API is included in **allow_methods**.

Configuration Parameters

Table 6-2 Configuration parameters

Parameter	Description
Allowed Origins	**Access-Control-Allow-Origin** response header, which specifies either a single origin, which tells browsers to allow that origin to access an API; or else — for requests without credentials — the "*" wildcard, to tell browsers to allow any origin to access the API. Separate multiple URIs using commas.
Allowed Methods	**Access-Control-Allow-Methods** response header, which specifies the HTTP methods allowed when accessing the API. Separate multiple methods using commas.
Allowed Headers	**Access-Control-Allow-Headers** response header, which specifies request headers that can be used when making an XMLHttpRequest. Separate multiple headers using commas. By default, simple request headers **Accept**, **Accept-Language**, **Content-Language**, and **Content-Type** (only if the value is **application/x-www-form-urlencoded**, **multipart/form-data**, or **text/plain**) are carried in requests. You do not need to configure these headers in this parameter. **NOTE** ● When you create a CORS policy, **Allowed Headers** is blank by default, which means cross-domain requests cannot carry any custom headers. ● Setting **Allowed Headers** to an asterisk (*) means cross-domain requests can carry any custom headers.

Parameter	Description
Exposed Headers	**Access-Control-Expose-Headers** response header, which specifies which response headers can be contained in the response of XMLHttpRequest. Separate multiple headers using commas. By default, basic response headers **Cache-Control**, **Content-Language**, **Content-Type**, **Expires**, **Last-Modified**, and **Pragma** can be contained in the response. You do not need to configure these headers in this parameter. **NOTE** ● When you create a CORS policy, **Exposed Headers** is blank by default, which means the JavaScript code of a browser cannot parse the headers in a cross-domain access response. However, the following basic response headers obtained using the getResponseHeader() method of the XMLHttpRequest object are excluded: **Cache-Control**, **Content-Language**, **Content-Type**, **Expires**, **Last-Modified**, and **Pragma**. ● Setting **Exposed Headers** to an asterisk (*) means the JavaScript code of a browser can parse all the headers in a cross-domain access response.
Maximum Age	**Access-Control-Max-Age** response header, which specifies for how many seconds the results of a preflight request can be cached. No more preflight requests will be sent within the specified period.
Allowed Credentials	**Access-Control-Allow-Credentials** response header, which specifies whether XMLHttpRequest requests can carry cookies.

Example Script

```
{
  "allow_origin": "*",
  "allow_methods": "GET,POST,PUT",
  "allow_headers": "Content-Type,Accept,Accept-Ranges,Cache-Control",
  "expose_headers": "X-Request-Id,X-Apig-Latency",
  "max_age": 86400,
  "allow_credentials": true
}
```

6.3 HTTP Response Header Management

HTTP response headers are part of the response returned by APIG to a client that calls an API. You can customize HTTP response headers that will be contained in an API response.

NOTE

If your gateway does not support this policy, contact technical support to upgrade the gateway to the latest version.

Usage Guidelines

- You have understood the **guidelines for policy creation and API binding**.
- You cannot modify the response headers (including **x-apig-*** and **x-request-id**) added by APIG or the headers required for CORS.

Configuration Parameters

Table 6-3 Configuration parameters

Parameter	Description
Name	Response header name, which is case-insensitive and must be unique within a plug-in. You can add a maximum of 10 response headers.
Value	Value of the response header. This parameter does not take effect and can be left blank if you set **Action** to **Delete**.

Parameter	Description
Action	Response header operation. You can override, append, delete, skip, or add response headers. **Override** ● The value of this response header will override the value of the same response header that exists in an API response. ● If an API response contains multiple response headers with the same name, only the value of this response header will be returned. ● If there is no response header with the same name in an API response, the value of this response header will be returned. **Append** ● If an API response contains the specified header, the value you set here will be added, following the existing value. The two values will be separated with commas (,). ● If an API response contains multiple response headers with the same name, values of these response headers will be returned and separated with commas (,), appended by the value of this response header. ● If there is no response header with the same name in an API response, the value of this response header will be returned. **Delete** ● This response header will be deleted if a response header with the same name exists in an API response. ● If an API response contains multiple response headers with the same name, all these response headers will be deleted. **Skip** ● This response header will be skipped if a response header with the same name exists in an API response. ● If an API response contains multiple response headers with the same name, values of all these response headers will be returned. ● If there is no response header with the same name in an API response, the value of this response header will be returned. **Add**

Parameter	Description
	The value of this response header will be returned in an API response even if the response contains a response header with the same name.

Example Script

```
{
   "response_headers": [
      {
         "name": "test",
         "value": "test",
         "action": "append"
      },
      {
         "name": "test1",
         "value": "test1",
         "action": "override"
      }
   ]
}
```

6.4 Request Throttling 2.0

A request throttling 2.0 policy limits the number of times that an API can be called within a specific time period. Parameter-based, basic, and excluded throttling is supported.

- Basic throttling

  Throttle requests by API, user, credential, or source IP address. This function is equivalent to a traditional request throttling policy (see **Request Throttling**) but is incompatible with it.
- Parameter-based throttling

  Throttle requests based on headers, path parameter, method, query strings, or system parameters.
- Excluded throttling

  Throttle requests based on specific credentials or tenants.

NOTE

If your gateway does not support this policy, contact technical support to upgrade the gateway to the latest version.

Usage Guidelines

- You have understood the **guidelines for policy creation and API binding**.
- A request throttling policy becomes invalid if a request throttling 2.0 policy is bound to the same API as the existing one.
- You can define a maximum of 100 parameter-based throttling rules. The parameter name can contain 1 to 32 characters.
- The policy content cannot exceed 65,535 characters.

Parameter Description

Table 6-4 Parameter description

Parameter	Description
Throttling	High-performance throttling is recommended. ● **High precision**: better for low concurrency scenarios (performance is affected) ● **High performance**: better for medium concurrency scenarios (performance is less affected, with small occasional errors) ● **Single node**: better for high concurrency scenarios (request throttling within each node; performance is least affected, with small occasional errors)
Policy Type	● API-specific Monitor and control the requests for a single API. ● API-sharing Monitor and control requests for all APIs bound with the policy.
Period	For how long you want to limit the number of API calls. This parameter can be used together with the following parameters: ● **Max. API Requests**: Limit the maximum number of times an API can be called within a specific period. ● **Max. User Requests**: Limit the maximum number of times an API can be called by a user within a specific period. ● **Max. Credential Requests**: Limit the maximum number of times an API can be called by a credential within a specific period. ● **Max. IP Address Requests**: Limit the maximum number of times an API can be called by an IP address within a specific period.
Max. API Requests	The maximum number of times each bound API can be called within the specified period. This parameter must be used together with **Period**.

Parameter	Description
Max. User Requests	The maximum number of times each bound API can be called by a user within the specified period. For APIs with IAM authentication, the throttling is based on a project ID; for APIs with app authentication, the throttling is based on an account ID. For details about account ID and project ID, see the description about **Excluded Tenants** in this table. ● The value of this parameter cannot exceed that of **Max. API Requests**. ● This parameter must be used together with **Period**. ● If there are many users under your account that access an API, the request throttling limits of the API will apply to all these users.
Max. Credential Requests	The maximum number of times each bound API can be called by a credential within the specified period. This limit only applies to APIs that are accessed through app authentication. ● The value of this parameter cannot exceed that of **Max. API Requests**. ● This parameter must be used together with **Period**.
Max. IP Address Requests	The maximum number of times each bound API can be called by an IP address within the specified period. You can configure the **real_ip_from_xff** parameter of the gateway to use the IP address in the **X-Forwarded-For** header as the basis for request throttling. ● The value of this parameter cannot exceed that of **Max. API Requests**. ● This parameter must be used together with **Period**.
Parameter-based Throttling	Enable or disable parameter-based throttling. After this function is enabled, API requests are throttled based on the parameters you set.
Parameters	Define parameters for rule matching. ● **Parameter Location**: the location of a parameter used for rule matching. – **path**: API request URI. This parameter is configured by default. – **method**: API request method. This parameter is configured by default. – **header**: the key of a request header. – **query**: the key of a query string. – **system**: a system parameter. ● **Parameter**: the name of a parameter to match the specified value in a rule.

Parameter	Description
Rules	Define throttling rules. A rule consists of conditions, an API request throttling limit, and a period. To add more rules, click **Add Rule**. ● Rule Click to set condition expressions. To set an expression, select a parameter and operator, and enter a value. – **=**: equal to – **!=**: not equal to – **pattern**: regular expression – **enum**: enumerated values. Separate them with commas (,). ● Max. API Requests The maximum number of times that an API can be called within a specific time period. ● Period A period of time that will apply with the throttling limit you set. If this parameter is not specified, the period set in the **Police Information** area will be used. For example, configure parameter-based throttling as follows: add the **Host** parameter and specify the location as **header**; add the condition **Host = www.abc.com**, and set the throttling limit to **10** and the period to 60s. For APIs whose **Host** parameter in the request header is equal to **www.abc.com**, they cannot be called again once called 10 times in 60s.
Excluded Throttling	Enable or disable excluded throttling. After this function is enabled, the throttling limits for excluded tenants and credentials override the **Max. User Requests** and **Max. Credential Requests** set in the **Basic Throttling** area.
Excluded Tenants	**Tenant ID**: an account ID or project ID. ● Specify a project ID for an API with app authentication. For details, see section "Obtaining a Project ID" in the *API Gateway API Reference*. ● Specify an account ID (not IAM user ID) for an API with IAM authentication. For details, see section "Obtaining an Account Name and Account ID" in the *API Gateway API Reference*. **Threshold**: the maximum number of times that a specific tenant can access an API within the specified period. The threshold cannot exceed the value of **Max. API Requests** in the **Basic Throttling** area.

Parameter	Description
Excluded Credentials	Select a credential, and specify the maximum number of times that the credential can access an API within the specified period. The threshold cannot exceed the value of **Max. API Requests** in the **Basic Throttling** area.

Example Script

```
{
  "scope": "basic",
  "default_interval": 60,
  "default_time_unit": "second",
  "api_limit": 100,
  "app_limit": 50,
  "user_limit": 50,
  "ip_limit": 20,
  "specials": [
    {
      "type": "app",
      "policies": [
        {
          "key": "e9230d70c749408eb3d1e838850cdd23",
          "limit": 10
        }
      ]
    },
    {
      "type": "user",
      "policies": [
        {
          "key": "878f1b87f71c40a7a15db0998f358bb9",
          "limit": 10
        }
      ]
    }
  ],
  "algorithm": "counter",
  "parameters": [
    {
      "id": "3wuj354lpptv0toe0",
      "value": "reqPath",
      "type": "path",
      "name": "reqPath"
    },
    {
      "id": "53h7e7j11u38l3ocp",
      "value": "method",
      "type": "method",
      "name": "method"
    },
    {
      "id": "vv502bnb6g40td8u0",
      "value": "Host",
      "type": "header",
      "name": "Host"
    }
  ],
  "rules": [
    {
      "match_regex": "[\"Host\",\"==\",\"www.abc.com\"]",
      "rule_name": "u8mb",
      "time_unit": "second",
      "interval": 2,
      "limit": 5
    }
  ]
 }
```

6.5 Kafka Log Push

Kafka log push policies push calling logs of open APIs to Kafka for analysis.

NOTE

If your gateway does not support this policy, contact technical support to upgrade the gateway to the latest version.

Usage Guidelines

- You have understood the **guidelines for policy creation and API binding**.
- A maximum of five Kafka log push policies can be created for a gateway.
- APIs bound with a Kafka log push policy will deteriorate in performance by 30%.

Configuration Parameters

Table 6-5 Parameter description

Parameter	Description
Policy Information	
Broker Address	Connection address of the target Kafka. Separate multiple addresses with commas (,).
Topic	Topic of the target Kafka to report logs to.
Key	Partition of Kafka for storing logs as an ordered message queue. If this parameter is left blank, logs are stored in different partitions.
Retry	Configuration for retrying when logs fail to be pushed to Kafka. ● **Retry Times**: the number of retry attempts in case of a failure. Enter 0 to 5. ● **Retry Interval**: the interval of retry attempts in case of a failure. Enter 1 to 10 seconds.
SASL Configuration	
Security Protocol	Protocol used for connecting to the target Kafka. ● **PLAINTEXT**: user authentication protocol of the default access point ● **SASL_PLAINTEXT**: SASL user authentication protocol ● **SASL_SSL**: SSL user authentication protocol

Parameter	Description
Message Tx/Rx Mechanism	Message transmission and receiving mechanism of the target Kafka. The default value is **PLAIN**.
SASL Username	This parameter is available only if **Security Protocol** is set to **SASL_PLAINTEXT** or **SASL_SSL**. Username used for SASL or SSL authentication.
SASL Password	This parameter is available only if **Security Protocol** is set to **SASL_PLAINTEXT** or **SASL_SSL**. User password used for SASL or SSL authentication.
Confirm SASL Password	This parameter is available only if **Security Protocol** is set to **SASL_PLAINTEXT** or **SASL_SSL**. Enter the SASL password again.
Certificate Content	This parameter is available only if **Security Protocol** is set to **SASL_SSL**. CA certificate used for SSL authentication.
Metadata Configuration	
System Metadata	System fields that need to be included in pushed logs. By default, the **start_time**, **request_id**, **client_ip**, **request_time**, **http_status**, **scheme**, **request_method**, **host**, **uri**, **upstream_addr**, **upstream_status**, **upstream_response_time**, **http_x_forwarded_for**, **http_user_agent**, and **error_type** fields are carried in logs. You can also specify other system fields that need to be included.
Request Data	API request information that needs to be included in pushed logs. ● **The log contains the request header**: Specify a header that needs to be included. Separate multiple headers with commas (,). The asterisk (*) can be used as a wildcard. ● **The log contains the request QueryString**: Specify a query string that needs to be included. Separate multiple query strings with commas (,). The asterisk (*) can be used as a wildcard. ● **The log contains the request body**: If this option is selected, logs will contain the body of API requests.

Parameter	Description
Response Data	API response information that needs to be included in pushed logs. ● **The log contains the response header**: Specify a header that needs to be included. Separate multiple headers with commas (,). The asterisk (*) can be used as a wildcard. ● **The log contains the response body**: If this option is selected, logs will contain the body of API request responses.
Customized Authentication	Custom authentication information that needs to be included in pushed logs. ● **Frontend**: Enter a response field of frontend authentication that needs to be included. Separate multiple fields with commas (,). ● **Backend**: Enter a response field of backend authentication that needs to be included. Separate multiple fields with commas (,).

6.6 Circuit Breaker

Circuit breaker policies protect your backend services when a performance issue occurs. If the backend service of an API times out for *N* consecutive times or if the latency is long, the downgrade mechanism of a circuit breaker policy is triggered to return an error to the API caller or forward requests to a specified backend. After the backend service recovers, the circuit breaker closes and requests become normal.

NOTE

If your gateway does not support this policy, contact technical support to upgrade the gateway to the latest version.

Prerequisites

You have understood the **guidelines for policy creation and API binding**.

Parameter Description

Table 6-6 Parameter description

Parameter	Description
Policy Type	● API-specific Control requests for a single API. ● API-sharing Control requests for all APIs bound with the policy.

Parameter	Description
Circuit Breaker Type	Triggering type of the circuit breaker. ● **Timeout downgrade**: The circuit breaker will be triggered upon backend timeout. ● **Condition downgrade**: The circuit breaker will be triggered when configured match conditions are met.
Condition Type	Triggering mode of the circuit breaker. ● **Count**: Once the number of requests that meet conditions within a specified time window reaches the threshold, the circuit breaker is immediately triggered. ● **Percentage**: Once the percentage of requests that meet conditions within a specified time window reaches the threshold, the circuit breaker is triggered after the time window expires.
Match Condition	This parameter is required only when **Circuit Breaker Type** is set to **Condition downgrade**. Configure triggering conditions for the circuit breaker. ● **Response Error Codes**: The circuit breaker will be triggered if the backend responds with specified status codes. ● **Response Latency**: The circuit breaker will be triggered if the backend response latency reached a specified threshold.
Time Window (s)	The period for determining how many times have the conditions been met. Use this parameter together with **Threshold** or **Min Percentage**. If the threshold or percentage is reached, the circuit breaker is triggered.

Parameter	Description
Threshold	This parameter is required only when **Condition Type** is set to **Count**. Set the threshold for triggering the circuit breaker. Use this parameter together with **Time Window**. Once the number of backend requests that meet the conditions within the time window reaches the threshold, the circuit breaker is triggered. **NOTE** A circuit breaker policy is triggered for a single gateway component. If your gateway has multiple components, the triggering for each component is determined separately. If the threshold is reached within the time window for a gateway component, requests sent to the component trigger the circuit breaker, and other gateway components still forward requests normally. A gateway component is a connection address of your gateway. To view the number of gateway components, go to the **Gateway Information** page of the gateway and view the number of IP addresses in **Private Network Access IP**.
Min Calls	This parameter is required only when **Condition Type** is set to **Percentage**. Set the minimum number of API calls that will trigger the circuit breaker within the time period. The circuit breaker will not be triggered if the number of API calls within the time period is less than this value.
Min Percentage (%)	This parameter is required only when **Condition Type** is set to **Percentage**. Set the threshold for triggering the circuit breaker. Use this parameter together with **Time Window**. Once the percentage of backend requests that meet the conditions within the time window reaches the threshold, the circuit breaker is triggered.
Control Duration (s)	Time for which the circuit breaker will be on. When the time is reached, the circuit breaker will be off.
Backend Downgrade	Determine whether to enable backend downgrade. ● Enable: Requests for APIs that have triggered a downgrade will be forwarded to a specified backend. ● Disable: Requests for APIs that have triggered a downgrade will not be forwarded to any backend. Instead, HTTP status code **503** and an error message indicating that the service is unavailable will be returned.

Parameter	Description
Backend Type	This parameter is required only when **Backend Downgrade** is enabled. Specify the backend type to which requests will be forwarded when the circuit breaker is on. ● **Mock**: The defined response will be returned. – **Status Code**: the status code to be included in the response – **Response**: the response body, which is in JSON format – **Response Header**: header parameters to be included in the response ● **HTTP&HTTPS**: Backend requests will be forwarded to a specified HTTP&HTTPS backend service. – **Load Balance Channel**: Determine whether to use a load balance channel to access the backend service. If yes, **create a load balance channel** in advance. – **Backend URL**: address of the backend service to forward requests to. – **Timeout (ms)**: backend request timeout. The default value is 5000 ms. ● **FunctionGraph**: Backend requests will be forwarded to a specified function. – **Function URN**: the unique identifier of a function. Click **Select** to select a function. – **Function Name**: automatically displayed after you select a function. – **Version/Alias**: version or alias of the function to be used to receive backend requests. – **Invocation Mode**: the mode in which the function is invoked. **Synchronous**: When receiving an invocation request, FunctionGraph immediately processes the request and returns a result. The client closes the connection once it has received a response from the backend. **Asynchronous**: After receiving an invocation request, FunctionGraph queues the request and returns the result after the request is successfully processed. The server processes the queuing requests one by one when it is idle. The client does not care about the invocation result. – **Timeout (ms)**: backend request timeout. The default value is 5000 ms.

Parameter	Description
	● **Passthrough**: Backend requests will be forwarded to the original API backend. To add header parameters to backend requests, click **Add Parameter**.
Downgrade Parameter Settings	Determine whether to enable downgrade parameter configuration. After this option is enabled, custom rules take precedence over the default triggering conditions and downgrade settings configured above. ● If a custom rule is matched, the triggering conditions and downgrade settings defined in the rule are applied. If the matched custom rule contains no triggering condition or downgrade settings, the default settings in **Trigger Configuration** and **Backend Downgrade** will be applied. ● If no custom rule is matched, the default settings will be applied.
Parameters	Define parameters for rule matching. ● **Parameter Location**: position of a parameter in API requests. ● **Parameter Name**: name of a parameter used for rule matching. By default, the system provides the **reqPath** (request path) and **method** (request method) parameters. Click **Add Parameter** to add parameters.

Parameter	Description
Rules	Customize matching rules for the circuit breaker. Click **Add Rule** to add rules. The system matches rules from top to bottom. Adjust the rule priority by moving the rules up or down. ● **Conditions**: Click ✎ to set condition expressions. If there are three or more expressions, you can layer them by clicking **Set Lower Level**. – **=**: equal to – **!=**: not equal to – **pattern**: regular expression – **enum**: enumerated values. Separate them with commas (,). ● For details about how to configure the triggering conditions and backend downgrade, see the instructions for the default settings above. Example: You have enabled **Downgrade Parameter Settings** and added rules **rule01** and **rule02** in sequence. And you have disabled **Trigger Configuration** and enabled **Backend Downgrade** for **rule01**, and have enabled both options for **rule02**. With these settings, the circuit breaker first checks whether the conditions of **rule01** are met. If yes, the circuit breaker is turned on based on the default settings because no triggering condition has been defined in **rule01**, and backend downgrade configured in **rule01** is executed. If no, the check is continued for **rule02**.

Example Script

```
{
  "breaker_condition":{
    "breaker_type":"timeout",
    "breaker_mode":"counter",
    "unhealthy_threshold":30,
    "time_window":15,
    "open_breaker_time":15,
    "unhealthy_percentage":51,
    "min_call_threshold":20
  },
  "scope":"share",
  "downgrade_default":{
    "type":"http",
    "passthrough_infos":null,
    "func_info":null,
    "mock_info":null,
    "http_info":{
      "isVpc":false,
      "vpc_channel_id":"",
      "address":"10.10.10.10",
      "scheme":"HTTP",
      "method":"GET",
      "path":"/demo",
      "timeout":5000
    },
    "http_vpc_info":null
  },
  "downgrade_parameters":[
  {
    "name":"reqPath",
    "type":"path",
    "value":"path",
    "disabled":true,
    "focused":true,
    "id":"92002eqbpilg6g"
  },
  {
    "name":"method",
    "type":"method",
    "value":"method",
    "disabled":true,
    "focused":true,
    "id":"tuvxetsdqvcos8"
  }],
  "downgrade_rules":[
  {
    "rule_name":"rule-test1",
    "parameters":[
      "reqPath",
      "method"
    ],
    "match_regex":"[\"reqPath\",\"==\",\"/test\"]",
    "downgrade_backend":{
      "type":"mock",
      "passthrough_infos":null,
      "func_info":null,
      "mock_info":{
        "status_code":200,
        "result_content":"{status: ok}",
        "headers":[]
      },
      "http_info":null,
      "http_vpc_info":null
    },
    "breaker_condition":{
      "breaker_type":"timeout",
      "breaker_mode":"percentage",
      "unhealthy_threshold":30,
      "time_window":15,
      "open_breaker_time":15,
      "unhealthy_percentage":51,
      "min_call_threshold":20
    }
  }]
}
```

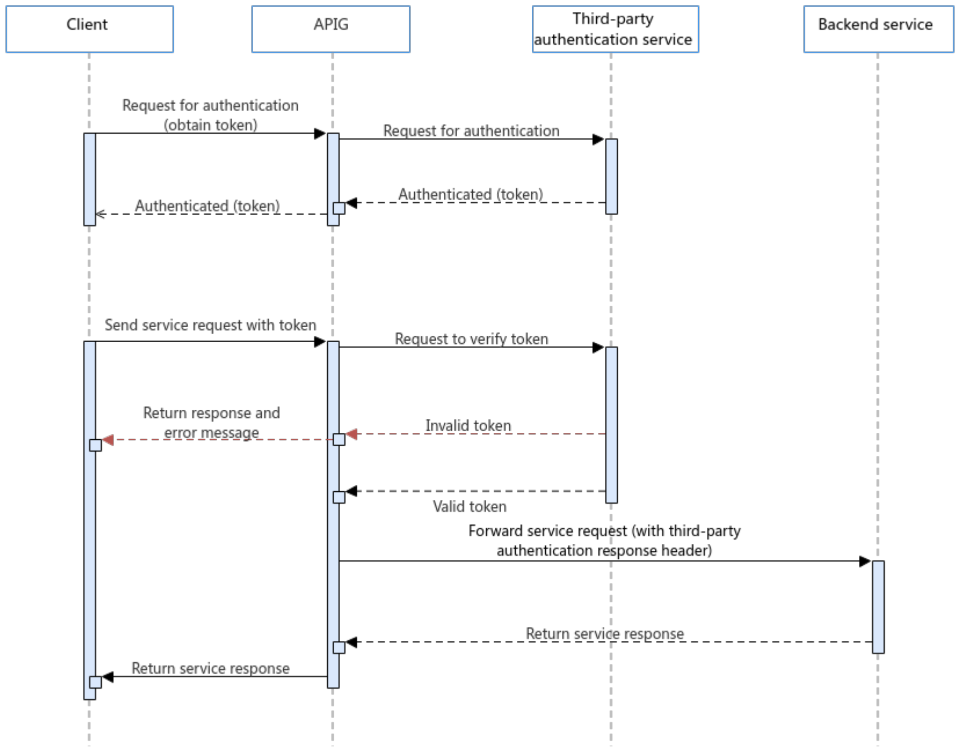
6.7 Third-Party Authorizer

You can configure your own service to authenticate API requests. APIG first invokes this service for authentication, and then invokes the backend service after receiving a success response.

NOTE

If your gateway does not support this policy, contact technical support to upgrade the gateway to the latest version.

The following figure shows the principle of third-party authentication. After binding a third-party authentication policy to an API, call the API by referring to **Calling APIs**.


Client
APIG
Third-party authentication service
Backend service
Request for authentication (obtain token)
Request for authentication
Authenticated (token)
Authenticated (token)
Send service request with token
Request to verify token
Return response and error message
Invalid token
Valid token
Forward service request (with third-party authentication response header)
Return service response
Return service response

Prerequisites

You have understood the **guidelines for policy creation and API binding**.

Configuration Parameters

Table 6-7 Configuration parameters

Parameter	Description
Load Balance Channel	Whether to connect a third-party authentication service using a load balance channel. ● **Configure**: Select a **load balance channel**. ● **Skip**: Enter the path of the authentication service.

Parameter	Description
Backend URL	● **Method** **GET**, **POST**, **PUT**, and **HEAD** are supported. ● **Protocol** **HTTP** or **HTTPS**. **HTTPS** is recommended for transmitting important or sensitive data. ● **Load Balance Channel** (if applicable) Set this parameter only if a load balance channel is used. Select a load balance channel. If no required channel is available, click **Create Load Balance Channel** to create one. ● **Backend Address** (if applicable) Set this parameter if no load balance channel is used. Enter the access address of the authentication service in the format of *Host:Port*. *Host* indicates the IP address or domain name for accessing the authentication service. If no port is specified, ports 80 and 443 are used by default for HTTP and HTTPS, respectively. **NOTE** Only IPv4 addresses are supported. ● **Path** Path (URL) of the authentication service.
Timeout (ms)	Timeout of the authentication service. It cannot exceed the max. timeout of the backend service. View the timeout limit on the **Parameters** tab of the gateway details page.
Host Header	**Set this parameter only if a load balance channel is used.** Define a host header for requests to be sent to cloud servers associated with the load balance channel. By default, the original host header in each request is used.
Brute Force Threshold	IP addresses whose number of third-party authentication failure attempts within 5 minutes exceeds this threshold will be blocked. They will be unblocked after 5 minutes. For example, if an IP address has failed third-party authentication more than the configured threshold in the third minute, the address is blocked, and will be unblocked after 2 minutes.
Identity Sources	Parameters to obtain from the original API requests for third-party authentication. Max. 10 headers and 10 query strings. If not specified, all headers and query strings in the original requests will be used.

Parameter	Description
Relaxed Mode	When this option is enabled, APIG accepts client requests even when your authentication service cannot connect or returns an error code starting with "5".
Allow Original Request Body	When this option is enabled, the original request body is included for authentication.
Request Body Size (bytes)	Available only when **Allow Original Request Body** is enabled. The value cannot exceed the max. request body size of the gateway. View the request body size limit on the **Parameters** tab of the gateway details page.
Allow Original Request Path	When this option is enabled, the original request path is added to the end of the authentication request path.
Return Response	When this option is enabled, the authentication response is returned on failure.
Allowed Response Headers	Headers to obtain from the authentication response and send to the backend service, when the authentication is successful. Max. 10 headers.
Simple Authentication	When this option is enabled, status codes starting with "2" indicate successful authentication.
Authentication Result	Available only when **Simple Authentication** is disabled. Responses whose headers contain these parameters with the same values indicate successful authentication.
Blacklist/Whitelist	When this option is enabled, whether API requests require third-party authentication depends on the configured blacklist or whitelist rules.
Type	● Whitelist API requests matching the whitelist rules do not require third-party authentication. ● Blacklist API requests matching the blacklist rules require third-party authentication.

Parameter	Description
Parameters	Define parameters for rule matching. ● **Parameter Location**: the location of a parameter used for rule matching. – **path**: API request URI. This parameter is configured by default. – **method**: API request method. This parameter is configured by default. – **header**: the key of a request header. – **query**: the key of a query string. – **system**: a system parameter. ● **Parameter**: the name of a parameter to match the specified value in a rule.
Rules	Define conditions for rule matching. Click **Add Rule** and edit the rule name and conditions. In the **Condition Expressions** dialog box, select a parameter and operator, and enter a value. ● **=**: equal to ● **!=**: not equal to ● **pattern**: regular expression ● **enum**: enumerated values. Separate them with commas (,).

Example Script

```
{
  "auth_request": {
    "method": "GET",
    "protocol": "HTTPS",
    "url_domain": "192.168.10.10",
    "timeout": 5000,
    "path": "/",
    "vpc_channel_enabled": false,
    "vpc_channel_info": null
  },
  "custom_forbid_limit": 100,
  "carry_body": {
    "enabled": true,
    "max_body_size": 1000
  },
  "auth_downgrade_enabled": true,
  "carry_path_enabled": true,
  "return_resp_body_enabled": false,
  "carry_resp_headers": [],
  "simple_auth_mode_enabled": true,
  "match_auth": null,
  "rule_enabled": false,
  "rule_type": "allow"
}
```

6.8 Request Throttling

Request throttling limits the number of times APIs can be called by a user or app within a specific time period to protect backend services. The throttling can be down to the minute or second. To ensure service continuity of an API, create a request throttling policy for the API.

Usage Guidelines

- You have understood the **guidelines for policy creation and API binding**.
- Adding a request throttling policy to an API means binding them to each other. An API can be bound with only one request throttling policy for a given environment, but each request throttling policy can be bound to multiple APIs.
- For APIs not bound with a request throttling policy, the throttling limit is the value of **ratelimit_api_limits** set on the **Parameters** page of the gateway.

Configuration Parameters

Table 6-8 Configuration parameters

Parameter	Description
Name	Request throttling policy name.
Type	API-based or API-shared request throttling. ● **API-specific**: Request throttling is based on every API to which the policy is bound. ● **API-sharing**: Request throttling is based on all APIs as a whole to which the policy is bound.
Period	For how long you want to limit the number of API calls. This parameter can be used together with the following parameters: ● **Max. API Requests**: Limit the maximum number of times an API can be called within a specific period. ● **Max. User Requests**: Limit the maximum number of times an API can be called by a user within a specific period. ● **Max. Credential Requests**: Limit the maximum number of times an API can be called by a credential within a specific period. ● **Max. IP Address Requests**: Limit the maximum number of times an API can be called by an IP address within a specific period.
Max. API Requests	The maximum number of times each bound API can be called within the specified period. This parameter must be used together with **Period**.

Parameter	Description
Max. User Requests	The maximum number of times each bound API can be called by a user within the specified period. **This limit only applies to APIs that are accessed through app or IAM authentication.** ● The value of this parameter cannot exceed that of **Max. API Requests**. ● This parameter must be used together with **Period**. ● If there are many users under your account that access an API, the request throttling limits of the API will apply to all these users.
Max. Credential Requests	The maximum number of times each bound API can be called by a credential within the specified period. **This limit only applies to APIs that are accessed through app authentication.** ● The value of this parameter cannot exceed that of **Max. User Requests** or **Max. API Requests**. ● This parameter must be used together with **Period**.
Max. IP Address Requests	Maximum times that an API can be requested by an IP address. You can configure the **real_ip_from_xff** parameter of the gateway to use the IP address in the **X-Forwarded-For** header as the basis for request throttling. ● The value of this parameter cannot exceed that of **Max. API Requests**. ● This parameter must be used together with **Period**.
Description	Description of the request throttling policy.

Follow-Up Operations

- To control the traffic of a credential, bind a request throttling policy to the credential by referring to **Binding a Request Throttling Policy to a Credential**. Traffic of the credential is limited by the excluded credential threshold, while traffic of APIs and users are still limited by the request throttling policy.
- To control the traffic of a tenant, bind a request throttling policy to the tenant by referring to **Binding a Request Throttling Policy to a Tenant**. Traffic of the tenant is limited by the excluded tenant threshold, while traffic of APIs and users are still limited by the request throttling policy.

Binding a Request Throttling Policy to a Credential

You have created a credential or obtained a credential ID from other tenants.

Step 1 On the request throttling policy details page, click the **Excluded Credentials** tab.

Step 2 Click **Select Excluded Credential**.

Step 3 Select a credential to exclude. You can use one of the following methods:

- To select an existing credential, click **Existing**, select a credential, and enter a threshold.
- To select a credential of other tenants, click **Cross-tenant**, and enter the credential ID and a threshold.

NOTE

Excluded credential thresholds take precedence over the value of **Max. Credential Requests**.

For example, a request throttling policy has been configured, with **Max. API Requests** being **10**, **Max. Credential Requests** being **3**, **Period** being 1 minute, and two excluded credentials (max. **2** API requests for credential A and max. **4** API requests for credential B). If the request throttling policy is bound to an API, credential A and B can access the API 2 and 4 times within 1 minute, respectively.

----End

Binding a Request Throttling Policy to a Tenant

Step 1 On the request throttling policy details page, click the **Excluded Tenants** tab.

Step 2 Click **Select Excluded Tenant**.

Step 3 Enter the tenant information.

Table 6-9 Excluded tenant configuration

Parameter	Description
Tenant ID	Account ID or project ID. For details, see the description about **Excluded Tenants** in **Table 6-4**.
Threshold	The maximum number of times an API can be called by the tenant within a specified period. The value of this parameter cannot exceed that of **Max. API Requests**.

Step 4 Click **OK**.

NOTE

Excluded tenant thresholds take precedence over the value of **Max. User Requests**.

For example, a request throttling policy has been configured, with **Max. API Requests** being **10**, **Max. User Requests** being **3**, **Period** being 1 minute, and two excluded tenants (max. **2** API requests for tenant A and max. **4** API requests for tenant B). If the request throttling policy is bound to an API, tenants A and B can access the API 2 and 4 times within 1 minute, respectively.

----End

6.9 Access Control

Access control policies are a type of security measures provided by APIG. You can use them to allow or deny API access from specific IP addresses, account names, or account IDs.

Access control policies take effect for an API only if they have been bound to the API.

Usage Guidelines

- You have understood the **guidelines for policy creation and API binding**.
- An API can be bound only with one access control policy of the same restriction type in an environment, but each access control policy can be bound to multiple APIs.

Configuration Parameters

Table 6-10 Configuration parameters

Parameter	Description
Name	Access control policy name.
Type	Type of the source from which API calls are to be controlled. ● **IP address**: Control API access by IP address. ● **Account name**: Control IAM authentication–based API access by account name, not IAM user name. Configure a single or multiple names separated by commas (,). Account name requirements: 1–64 characters, no commas (,) or all digits. The total length cannot exceed 1024 characters. ● **Account ID**: Control IAM authentication–based API access by account ID, not IAM user ID. Configure a single or multiple account IDs separated by commas (,). Each account ID contains 32 characters (letters and digits), separated by commas (,). Max. 1,024 characters. **NOTE** ● An API can be bound to two types of access control policies: account name and account ID. If both a blacklist and whitelist exist, API requests are verified only against the whitelist. If only a blacklist or whitelist exists, the account name and account ID verification results follow the AND logic. ● An API can be bound to three types of access control policies: IP address, account name, and account ID. IP addresses and accounts are in the AND relationship. Failure in verifying either of them will result in an API access failure. The same judgment logic applies to an API whether it is bound with a policy that controls access from specific IP address and account names or from specific IP addresses and account IDs.

Parameter	Description
Effect	Options: **Allow** and **Deny**. Use this parameter along with **Type** to control access from certain IP addresses, account names, or account IDs to an API.
IP Addresses	Required only when **Type** is set to **IP address**. IP addresses and IP address ranges that are allowed or not allowed to access an API. **NOTE** You can set a maximum of 100 IP addresses respectively to allow or deny access.
Account Names	Required only when **Type** is set to **Account name**. Enter the account names that are allowed or forbidden to access an API. Use commas (,) to separate multiple account names. Click the username in the upper right corner of the console and choose **My Credentials** to obtain the account name.
Account ID	Required only when **Type** is set to **Account ID**. Enter the account IDs that are allowed or forbidden to access an API. Use commas (,) to separate multiple account IDs. Click the username in the upper right corner of the console and choose **My Credentials** to obtain the account ID.

6.10 Signature Keys

Signature keys are used by backend services to verify the identity of APIG.

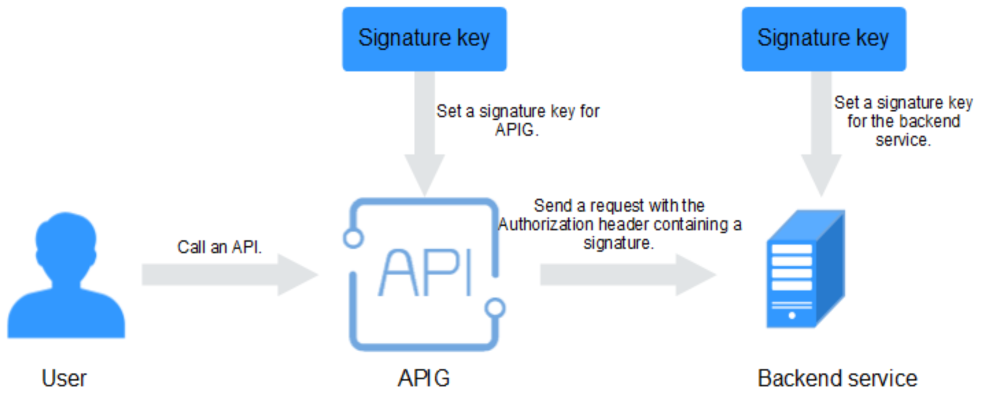
A signature key consists of a key and secret, and can be used only after being bound to an API. When an API bound with a signature key is called, APIG adds signature details to the API request. The backend service of the API signs the request in the same way, and verifies the identity of APIG by checking whether the signature is consistent with that in the **Authorization** header sent by APIG.

Usage Guidelines

- You have understood the **guidelines for policy creation and API binding**.
- An API can only be bound with one signature key in a given environment, but each signature key can be bound to multiple APIs.

Procedure

Figure 6-1 Signature key process flow


Signature key
Signature key
Set a signature key for APIG.
Set a signature key for the backend service.
Call an API.
Send a request with the Authorization header containing a signature.
API
User
APIG
Backend service

1. Create a signature key on the APIG console.
2. Bind the signature key to an API.
3. APIG sends signed requests containing a signature in the **Authorization** header to the backend service. The backend service can use different programming languages (Java, Go, Python, JavaScript, C#, PHP, C++, and C) to sign each request, and check whether the two signatures are consistent.

Configuration Parameters

Table 6-11 Configuration parameters

Parameter	Description
Name	Signature key name.
Type	Authentication type. Options: **HMAC**, **Basic auth**, **AES**.
Signature Algorithm	Select an AES signature algorithm. Options: ● aes-128-cfb ● aes-256-cfb
Key	Set the key based on the signature key type you have selected. ● If **Type** is **HMAC**, enter the key of the key pair used for app authentication. ● If **Type** is **Basic auth**, enter the username used for basic authentication. ● If **Type** is set to **AES**, enter the key used for AES authentication.

Parameter	Description
Secret	Enter the secret information based on the key type you have selected. ● If **Type** is **HMAC**, enter the secret of the key pair used for app authentication. ● If **Type** is **Basic auth**, enter the password used for basic authentication. ● If **Type** is set to **AES**, enter the vector used for AES authentication.
Confirm Secret	Enter the secret again.

Verifying the Signing Result

Sign each backend request by following the instructions in section "Creating Signatures for Backend Requests" in the *API Gateway Developer Guide*, and check whether the backend signature is consistent with the signature in the **Authorization** header of the API request.

6.11 Custom Authorizers

APIG supports custom authentication of both frontend and backend requests.

- Frontend custom authentication: If you already have an authentication system, you can configure it in a function and then create a custom authorizer by using the function to authenticate API requests.
- Backend custom authentication: You can create a custom authorizer to authenticate requests for different backend services, eliminating the need to customize APIs for different authentication systems and simplifying API development. You only need to create a function-based custom authorizer in APIG to connect to your backend authentication system.

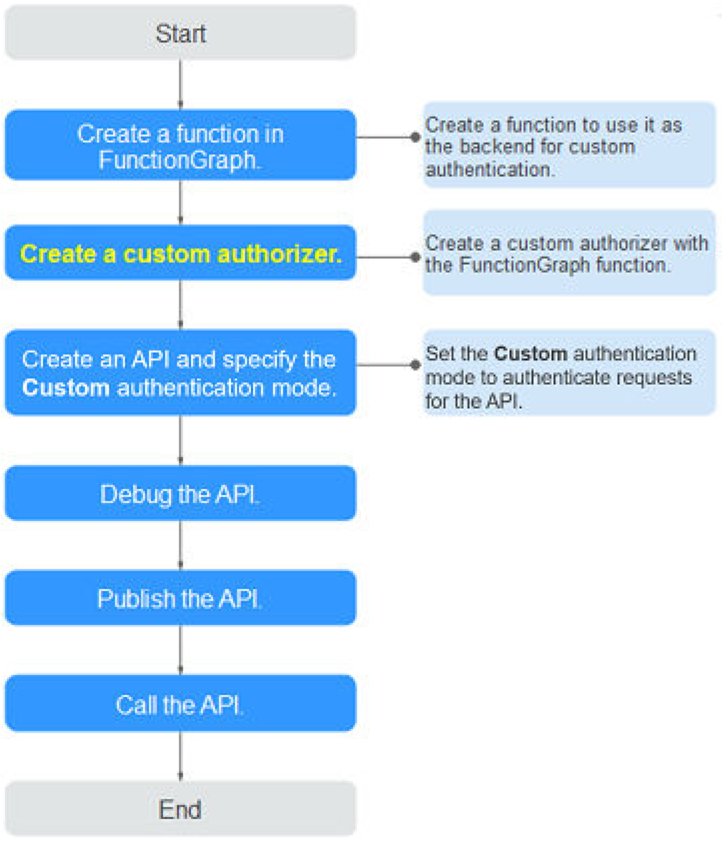
NOTE

Custom authentication is implemented using FunctionGraph and not supported if FunctionGraph is unavailable in the selected region.

For details about custom authentication, see the relevant section in the *API Gateway Developer Guide*.

The following figure shows the process of calling APIs through custom authentication.

Figure 6-2 Calling APIs through custom authentication


Start
Create a function in FunctionGraph.
Create a function to use it as the backend for custom authentication.
Create a custom authorizer.
Create a custom authorizer with the FunctionGraph function.
Create an API and specify the Custom authentication mode.
Set the Custom authentication mode to authenticate requests for the API.
Debug the API.
Publish the API.
Call the API.
End

Prerequisites

You have created a function in FunctionGraph.

Creating a Custom Authorizer

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **API Management** > **API Policies**.

Step 4 On the **Custom Authorizers** page, click **Create Custom Authorizer**.

Configure custom authorizer parameters.

Table 6-12 Parameters for creating a custom authorizer

Parameter	Description
Name	Authorizer name.
Type	● **Frontend**: Authenticates access to APIs. ● **Backend**: Authenticates access to backend services.
Function URN	Select a FunctionGraph function.

Parameter	Description
Version/Alias	Select a function version or alias. For details, see sections "Managing Versions" and "Managing Aliases" in the *FunctionGraph User Guide*.
Max. Cache Age (s)	The time for caching authentication results. The value ranges from 0s to 3,600s. **0** indicates that authentication results will not be cached.
Identity Sources	Request parameters used for authentication. This parameter is mandatory only if you set **Type** to **Frontend**, and **Max. Cache Age (s)** is greater than **0**. When the cache is used, this parameter is used as a search criterion to query authentication results.
Send Request Body	Determine whether to send the body of each API request to the authentication function. If you enable this option, the request body will be sent to the authentication function in the same way as the headers and query strings.
User Data	Customized request parameters to be used together with **Identity Sources** when APIG invokes a function.

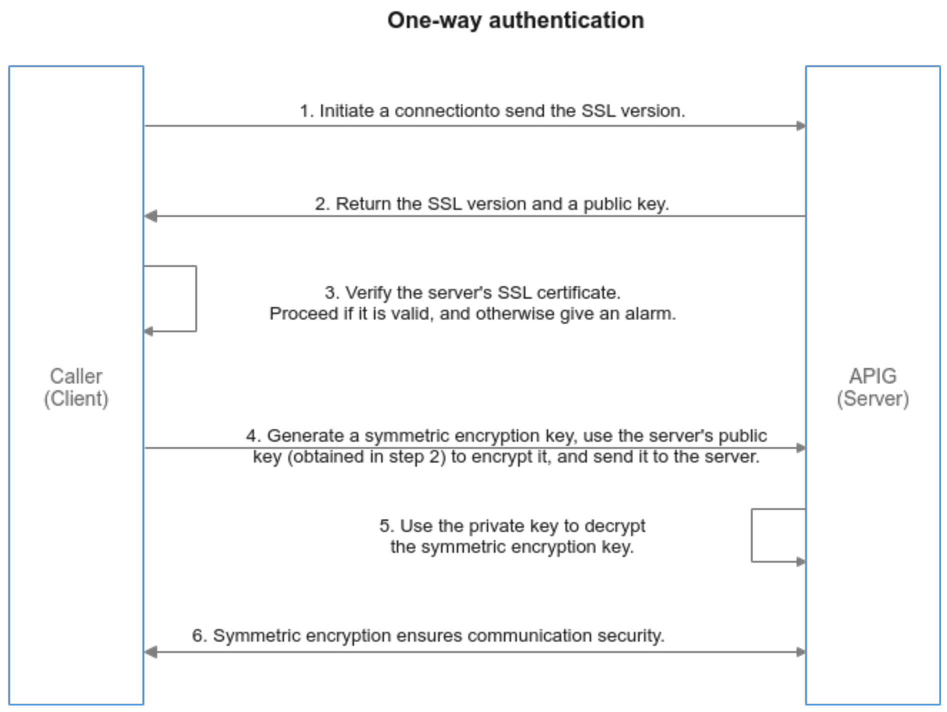
Step 5 Click **OK**.

----End

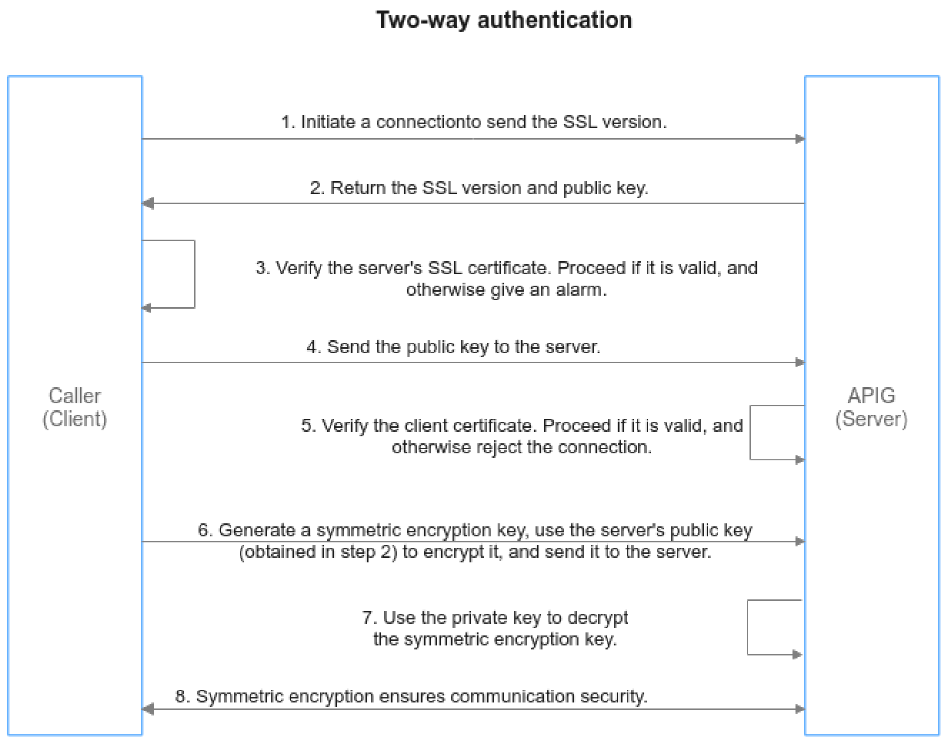
6.12 SSL Certificates

API groups that contain HTTPS-compatible APIs must have their independent domain names bound with SSL certificates. SSL certificates are used for data encryption and identity verification.

- One-way authentication: When connecting to the server, a client verifies whether the server is correct.


One-way authentication
Caller (Client)
APIG (Server)
1. Initiate a connectionto send the SSL version.
2. Return the SSL version and a public key.
3. Verify the server's SSL certificate. Proceed if it is valid, and otherwise give an alarm.
4. Generate a symmetric encryption key, use the server's public key (obtained in step 2) to encrypt it, and send it to the server.
5. Use the private key to decrypt the symmetric encryption key.
6. Symmetric encryption ensures communication security.

- Two-way authentication: When connecting to a server, a client verifies the server and the server also verifies the client.


Two-way authentication
Caller (Client)
APIG (Server)
1. Initiate a connectionto send the SSL version.
2. Return the SSL version and public key.
3. Verify the server's SSL certificate. Proceed if it is valid, and otherwise give an alarm.
4. Send the public key to the server.
5. Verify the client certificate. Proceed if it is valid, and otherwise reject the connection.
6. Generate a symmetric encryption key, use the server's public key (obtained in step 2) to encrypt it, and send it to the server.
7. Use the private key to decrypt the symmetric encryption key.
8. Symmetric encryption ensures communication security.

Prerequisites

- Only SSL certificates in PEM format are supported.
- SSL certificates support only the RSA, ECDSA, and DSA encryption algorithms.

Adding an SSL Certificate

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **API Management** > **API Policies**.

Step 4 On the **SSL Certificates** tab, click **Create SSL Certificate**.

Table 6-13 SSL certificate configuration

Parameter	Description
Name	Enter an SSL certificate name that conforms to specific rules to facilitate search.
Gateways Covered	● **Current**: The certificate will be displayed only for the current gateway. ● **All**: The certificate will be displayed for all gateways.
Content	SSL certificate content in PEM format. Open the target PEM certificate file using Notepad or other tools, and copy the certificate content to **Content**. If the certificate is not in PEM format, **convert it to this format**.
Key	SSL certificate key in PEM format. Open the KEY or PEM private key file using Notepad or other tools, and copy the private key to **Key**.
CA	For two-way authentication, you need to enter the CA certificate to verify both the server and client certificates. After the CA certificate is uploaded, the independent domain name needs to be bound to an **SSL certificate** to enable two-way authentication. Open the CA certificate file (.pem format) corresponding to the preceding certificate content as a text file and copy the CA content to **CA**. If the certificate is not in PEM format, **convert it to this format**. **NOTE** If your gateway does not support CA certificates, contact customer service to upgrade the gateway.

Step 5 Click **OK**. The SSL certificate is added.

----End

Converting Certificate Format to PEM

Format	Converting with OpenSSL
CER/CRT	Rename the certificate file **cert.crt cert.pem**.
PFX	● Run the private key export command. For example, run the following command to convert **cert.pfx** into **key.pem**: openssl pkcs12 -in cert.pfx -nocerts -out key.pem ● Run the certificate export command. For example, run the following command to convert **cert.pfx** into **cert.pem**: openssl pkcs12 -in cert.pfx -nokeys -out cert.pem
P7B	1. Run the certificate conversion command. For example, run the following command to convert **cert.p7b** into **cert.cer**: openssl pkcs7 -print_certs -in cert.p7b -out cert.cer 2. Rename the certificate file **cert.cer cert.pem**.
DER	● Run the private key export command. For example, run the following command to convert **privatekey.der** into **privatekey.pem**: openssl rsa -inform DER -outform PEM -in privatekey.der -out privatekey.pem ● Run the certificate export command. For example, run the following command to convert **cert.cer** into **cert.pem**: openssl x509 -inform der -in cert.cer -out cert.pem

Updating an SSL Certificate

On the certificate list page, locate the certificate to be updated, click **Modify** in the **Operation** column, and modify the certificate information.

- Updating the SSL certificate does not affect API calling.
- If the certificate to be updated has been bound to an independent domain name, all clients that access the domain name can view the updated certificate.
- If the updated SSL certificate has been bound to an independent domain name, the client authentication (HTTPS two-way authentication) is disabled by default when a CA certificate is added to the updated content.

Follow-Up Operations

After creating a certificate, **bind it** to an independent name of an API group.

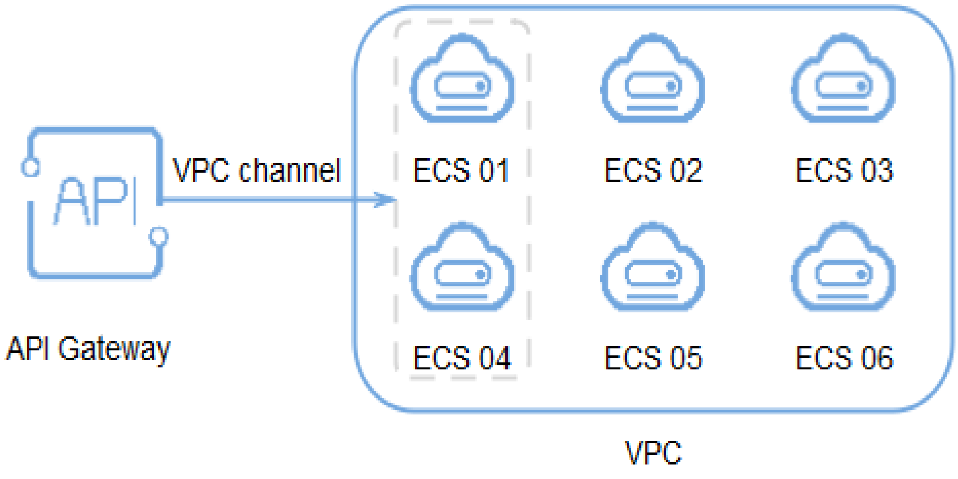
6.13 Load Balance Channels

Load balance channels expose your services through **dedicated gateways**, and are accessed through subnets in VPCs for lower latency. The server channel balances loads of backend services while the microservice channel automatically synchronizes service node changes.

After creating a load balance channel, you can configure it for an API of an HTTP&HTTPS backend service.

For example, six ECSs have been deployed, and a load balance channel has been created to reach ECS 01 and ECS 04. In this situation, APIG can access these two ECSs through the channel.

Figure 6-3 Accessing ECSs in a load balance channel through APIG



Prerequisites

- You have the **VPC Administrator** permission.
- To configure a server channel, ensure that you have created cloud servers that can communicate with APIG.
- To configure a Cloud Container Engine (CCE) microservice channel, ensure that you have created a cluster (a CCE cluster of VPC network model or a Turbo cluster) and a workload. For details, see section "Creating a CCE Standard/Turbo Cluster" or "Creating a Workload" in the *CCE User Guide*.

NOTICE

- If your gateway does not support microservice channels, contact technical support to upgrade the gateway to the latest version.
- The CCE cluster and the target gateway must be in the same VPC or connected to each other using a VPC peering connection. If the network is connected through the same VPC (with extended network segments) or a VPC peering connection, you need to add the container CIDR block of the cluster to **Routes** on the gateway details page.
- The workload must have a pod label configured. This label will be used to identify the workload, for example, a specific version of the workload, during **microservice configuration**. For details, see section "Labels and Annotations" in the *CCE User Guide*.
  - Configure a pod label when you create a workload by clicking **Create Workload**. On the workload creation page, in the **Advanced Settings** > **Labels and Annotations** > **Pod Label** area, configure the **app** label.
  - Configure a pod label when you create a workload by creating a YAML file. For example: **app=service01**.

```
spec:
  replicas: 2
  selector:
    matchLabels:
      app: 'service01'
```

Creating a Load Balance Channel

Step 1 Go to the APIG console.

Step 2 Select a gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **API Management** > **API Policies**.

Step 4 Click the **Load Balance Channels** tab.

Step 5 Click **Create Load Balance Channel** and configure basic information.

Table 6-14 Basic information

Parameter	Description
Name	Channel name.
Port	The host port of the channel, that is, the port of your backend services. Range: 1–65535

Parameter	Description
Routing Algorithm	The algorithm to be used to forward requests to cloud servers you select. The following routing algorithms are available: ● **WRR**: weighted round robin ● **WLC**: weighted least connection ● **SH**: source hashing ● **URI hashing**
Type	● **Server**: API requests will be distributed to ECSs or specified server IP addresses in the channel. For details, see **Step 6**. ● **Microservice**: API requests will be distributed to microservice IP addresses in the channel. For details, see **Step 7**.

Step 6 Configure servers if **Type** is set to **Server**.

NOTE

Load balance channels support private network load balancers. You can specify server addresses.

- Select cloud servers
  a. Click **Create Server Group**.
     In the displayed dialog box, enter server group information and click **OK**.

     Table 6-15 Server group parameters

Parameter	Description
Group Name	Enter a server group name. Using naming rules facilitates future search.
Weight	Enter the weight of the server group. The larger the weight, the more requests can be forwarded to the servers in the group.
Description	Enter a brief description of the server group.

  b. Click **Add Cloud Server**.
     In the displayed dialog box, select a subnet, select the cloud servers to be added, and click **OK**.
  c. After the configuration is complete, **configure health check**.
- Specify IP addresses
  a. Click **Create Server Group**.
     In the displayed dialog box, enter server group information and click **OK**. Configure parameters according to **Table 6-15**.

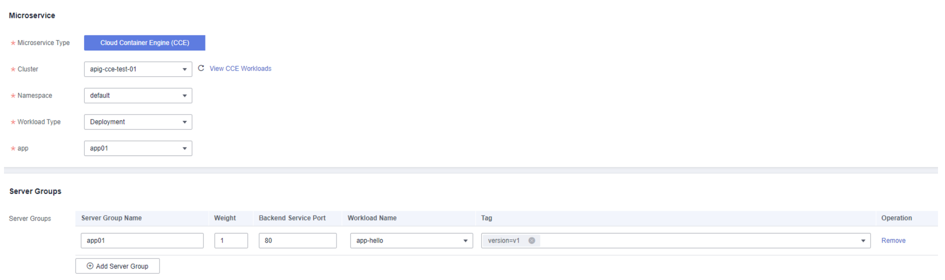
b. Click **Add Backend Server Address** and enter a backend server address.

Table 6-16 Backend server parameters

Parameter	Description
Backend Server Address	Backend server IP address.
Standby Node	If you enable this option, the backend server serves as a standby node. It works only when all non-standby nodes are faulty.
Port	Access port number of the backend server. If the port number is **0**, the port of the load balance channel is used. The port number ranges from 0 to 65535.
Server Status	Specify whether to enable the server. Requests are distributed to the server only if it is enabled.

c. After the configuration is complete, **configure health check**.

Step 7 Configure the microservice and server groups if **Type** is set to **Microservice**.



Microservice

* Microservice Type	Cloud Container Engine (CCE)
* Cluster	apig-cce-test-01
* Namespace	default
* Workload Type	Deployment
* app	app01

Server Groups

Server Group Name	Weight	Backend Service Port	Workload Name	Tag	Operation
app01	1	80	app-hello	version=v1	Remove

1. Configure microservice information according to the following table.

Table 6-17 CCE microservice configuration

Parameter	Description
Microservice Type	Fixed as **Cloud Container Engine (CCE)**.
Cluster	Select a cluster. Click **View CCE Console** to view the available clusters.
Namespace	Namespace of the cluster, which is an abstract collection of resources and objects.

Parameter	Description
Workload Type	– **Deployment**: Deployments do not store any data or status while they are running. – **StatefulSet**: StatefulSets store data and statuses while they are running. – **DaemonSet**: DaemonSets ensure that only one pod runs on all or some nodes. When a node is added to a cluster, a new pod is also added for the node. When a node is removed from a cluster, the pod is also reclaimed. **NOTE** If a DaemonSet is deleted, all pods created by it will be deleted. For details about workload types, see section "Workload Overview" in the *CCE User Guide*.
Service Label Key	Pod label of a workload. The service label name is the pod label key and the service label value is the pod label value. For details about pod labels, see section "Labels and Annotations" in the *CCE User Guide*.
Service Label Value	

2. Configure a server group.

   Click **Add Server Group** and set required parameters.

Table 6-18 Server group configuration of CCE microservice

Parameter	Description
Server Group Name	Same as the service label value by default. Modify the name if necessary.
Weight	Default value: **1**; range: 0–100. **NOTE** If **Routing Algorithm** is set to **URI hashing**, the weight is **1** by default and cannot be changed.
Backend Service Port	The port used by the backend server. If no port number is specified or the port number is **0**, the port of the load balance channel is used by default. The port number ranges from 0 to 65535.
Workload Name	Select a CCE workload.

Parameter	Description

Tag	Pod label of a workload. If a workload cannot be identified by certain service label name and value, select another pod label to specify the workload.
	For example, workloads **01** and **02** have the same **app** label, but they can be identified using the **version** or **test_name** tag.
	Workload **01**
	`spec:`
	`  replicas: 2`
	`  selector:`
	`    matchLabels:`
	`      app: 'app01'`
	`      version: 'v1'`
	Workload **02**
	`spec:`
	`  replicas: 2`
	`  selector:`
	`    matchLabels:`
	`      app: 'app01'`
	`      test_name: 'test_value'`

3. After the configuration is complete, **configure health check**.

Step 8 Configure health checks.

Table 6-19 Basic information

Parameter	Description
Protocol	The protocol used to perform health checks on cloud servers associated with the channel. Options: ● TCP ● HTTP ● HTTPS Default value: **TCP**.
Two-Way Authentication	**Set this parameter only when Protocol is set to HTTPS.** Determine whether to allow APIG to authenticate the API backend service. For details about how to configure the certificate for two-way authentication, see **Procedure**.
Path	**Set this parameter only when Protocol is not set to TCP.** The destination path for health checks.
Method	● GET ● HEAD

Parameter	Description
Check Port	The destination port for health checks. If this parameter is not specified, the port of the load balance channel is used by default.
Healthy Threshold	The number of consecutive successful checks required for a cloud server to be considered healthy. Range: 2–10. Default value: **2**
Unhealthy Threshold	The number of consecutive failed checks required for a cloud server to be considered unhealthy. Range: 2–10. Default value: **5**.
Timeout (s)	The timeout used to determine whether a health check has failed. Unit: s. Range: 2–30. Default value: **5**.
Interval (s)	The interval between consecutive checks. Unit: s. Range: 5–300. Default value: **10**.
Response Codes	**Set this parameter only when Protocol is not set to TCP.** The HTTP codes used to check for a successful response from a target.

Step 9 Click **Finish**.

For a microservice channel, adding, deleting, or modifying a pod IP address of the CCE workload will also change the backend server address of the channel.

----End

Follow-Up Operations

1. Ensure that a route has been added to the gateway. **To connect a CCE workload to a gateway through the same VPC (with extended network segments) or a VPC peering connection, you need to add a route.**
   a. Log in to the CCE console, choose **Clusters**, and click the name of the created CCE cluster.
   b. In the **Networking Configuration** area on the **Cluster Details** page, view and record the container CIDR block.
   c. Log in to the APIG console and click the gateway name on the **Gateways** page.
   d. In the **Routes** area on the **Gateway Information** page, check whether the added route is consistent with the container CIDR block. If not, add the correct route.
2. **Create APIs** to expose backend services deployed in the workload.

6.14 Managing Environments

An API can be called in different environments, such as production, testing, and development environments. RELEASE is the default environment provided by APIG.

Creating an Environment

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **API Management** > **API Policies**.

Step 4 Click the **Environments** tab.

Step 5 Click **Create Environment** and set the environment information.

Table 6-20 Environment information

Parameter	Description
Name	Environment name.
Description	Description of the environment.

Step 6 Click **OK**.

After the environment is created, it is displayed in the environment list.

----End

Accessing an Environment

You can call an API in the RELEASE environment by using a RESTful API. To access the API in other environments, add the **X-Stage** header to the request to specify an environment name. For example, add **X-Stage:DEVELOP** to the request header to access an API in the **DEVELOP** environment.

NOTE

APIG does not support API debugging with environment variables.

Follow-Up Operations

After creating an environment, **publish APIs** in the environment so that they can be called by API callers.

7 Credentials

7.1 Creating a Credential and Binding It to APIs

For APIs that use app authentication, create credentials to generate credential IDs and key/secret pairs. When calling such an API, bind a credential to the API, use the key/secret pair to replace that in the SDK so that APIG can authenticate your identity. For details about app authentication, see the *API Gateway Developer Guide*.

NOTE

- APIs that use IAM, custom authentication or require no authentication do not need credentials.
- You can create a maximum of 50 credentials for each gateway.

Creating a Credential

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **API Management** > **Credentials**.

Step 4 Click **Create Credential** and set credential information.

Table 7-1 Credential information

Parameter	Description
Name	Credential name.
Description	Description about the credential.

NOTE

You can customize AppKeys (keys) and AppSecrets (secrets). An AppKey is an automatically generated identifier, which is globally unique. You are not advised to customize one unless it is necessary.

Step 5 Click **OK**.

- After the credential is created, its name and ID are displayed on the **Credentials** page.
- Click the credential name and view the key and secret.

----End

Binding a Credential to APIs

Step 1 On the **Credentials** page, click the name of the target credential.

Step 2 In the **APIs** area, click **Bind to APIs**.

Step 3 Select an environment, API group, and APIs.

Step 4 Click **OK**.

To unbind an API, click **Unbind** in the row that contains the API.

NOTE

A credential can be bound to multiple APIs that use app authentication, and each such API can be bound with multiple credentials.

----End

7.2 Resetting Secret

Reset the secret of a credential as necessary. After resetting, the original secret becomes invalid and APIs to which the credential is bound cannot be called. To call the APIs, update the secret in the SDK. The key is unique and cannot be reset.

Procedure

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **API Management** > **Credentials**.

Step 4 Click the name of the target credential.

Step 5 Click **Reset Secret**.

Step 6 Click **OK**.

----End

7.3 Adding an AppCode for Simple Authentication

AppCodes are identity credentials of a credential used to call APIs in simple authentication mode. In this mode, the **X-Apig-AppCode** parameter (whose value is an AppCode on the credential details page) is added to the HTTP request header for quick response. APIG verifies only the AppCode and the request content does not need to be signed.

When an API is called using app authentication and simple authentication is enabled for the API, the key and secret can be used to sign and verify the API request. AppCodes can also be used for simple authentication.

NOTE

- For security purposes, simple authentication only supports API calls over HTTPS or gRPCS.
- You can create a maximum of five AppCodes for each credential.

Generating an AppCode

Step 1 Go to the APIG console.

Step 2 Select a dedicated gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **API Management** > **Credentials**.

Step 4 Click the name of the target credential.

Step 5 Under **AppCodes**, click **Add AppCode**.

Step 6 Configure AppCode information and click **OK**.

Table 7-2 AppCode configuration

Parameter	Description
AppCode Type	Select the method for generating an AppCode. ● **Automatically generated**: An AppCode is generated by the system. ● **Custom**: Specify an AppCode.
AppCode	Enter an AppCode if you set **AppCode Type** to **Custom**.

----End

Using AppCode for Simple Authentication of API Requests

Step 1 When creating an API, set **Authentication Mode** to **App** and enable **Simple Authentication**.

NOTE

After you enable simple authentication for an existing API, you need to publish the API again to make the configuration take effect.

Step 2 Bind a credential to the API.

Step 3 When sending a request, add the **X-Apig-AppCode** parameter to the request header and omit the request signature.

For example, when using curl, add the **X-Apig-AppCode** parameter to the request header and set the parameter value to the generated AppCode.

```
curl -X GET "https://api.exampledemo.com/testapi" -H "content-type: application/json"  -H "host: 
api.exampledemo.com" -H "X-Apig-AppCode: xhrJVJKABSOxc7d***********FZL4gSHEXkCMQC"
```

----End

7.4 Binding a Credential Quota Policy

A credential quota policy limits the number of API calls that a credential can make during a specified period.

Procedure

Step 1 Go to the APIG console.

Step 2 Select a gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **API Management** > **Credentials**.

Step 4 Click the name of the target credential.

Step 5 In the **Credential Quota Policies** area, click **Bind**.

Step 6 Specify the policy type.

- **Existing policy**: Select a policy.
- **New policy**: Configure a policy by referring to Table 7-3.

Table 7-3 Credential quota policy configuration

Parameter	Description
Name	Enter a credential quota policy name that conforms to specific rules to facilitate search.
Effective On	Time when the quota policy takes effect. For example, if **Effective On** is set to **Aug 8, 2020 05:05:00** and **Period** is set to 1 hour, the quota policy takes effect on Aug 8, 2020 05:05:00. The period from the fifth minute of an hour to the fifth minute of the next hour is a cycle, for example, 05:05:00-06:05:00.

Parameter	Description
Period	Period in which the quota policy is applied. The unit can be second, minute, hour, or day. This parameter must be used along with **Max. API Requests** to limit the total number of times an API can be called by a client within the specified period.
Max. API Requests	The maximum number of times that an API can be called by a client. This parameter must be used along with **Period**.
Description	Description about the credential quota policy.

Step 7 After the configuring is complete, click **OK**.

----End

7.5 Binding an Access Control Policy

As a protection mechanism for backend services, access control policies control the client (API caller) IP addresses that can access APIs. You can bind an access control policy to allow or deny access of specified IP addresses to an API.

Procedure

Step 1 Go to the APIG console.

Step 2 Select a gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **API Management** > **Credentials**.

Step 4 Click the name of the target credential.

Step 5 In the **Access Control Policy** area, click **Bind**.

Step 6 Configure the policy information.

Table 7-4 Access control policy configuration

Parameter	Description
Effect	Access control type. Options: ● **Allow**: Only clients with specified IP addresses are allowed to call APIs to which the credential is bound. ● **Deny**: Clients with specified IP addresses are not allowed to call APIs to which the credential is bound.
IP Addresses	Click **Add IP Address** to add IP addresses.

Step 7 After the configuring is complete, click **OK**.

----End

8 Monitoring & Analysis

8.1 API Monitoring

8.1.1 Monitoring Metrics

Introduction

This section describes the metrics that APIG reports to the Cloud Eye service. You can view metrics and alarms by using the Cloud Eye console.

Namespace

SYS.APIC

Metrics

Table 8-1 Metric description

Metric ID	Metric Name	Description	Value Range	Monitored Object and Dimension	Monitoring Period (Minute)
requests	Requests	Number of times that all APIs in a dedicated gateway have been called.	≥ 0	Monitored object: dedicated API gateway Dimension: instance_id	1

Metric ID	Metric Name	Description	Value Range	Monitored Object and Dimension	Monitoring Period (Minute)
error_4xx	4xx Errors	Number of times that all APIs in the dedicated gateway return a 4xx error.	≥ 0	Monitored object: dedicated API gateway Dimension: instance_id	1
error_5xx	5xx Errors	Number of times that all APIs in the dedicated gateway return a 5xx error.	≥ 0	Monitored object: dedicated API gateway Dimension: instance_id	1
throttled_calls	Throttled API Calls	Number of times that all APIs in the dedicated gateway have been throttled.	≥ 0	Monitored object: dedicated API gateway Dimension: instance_id	1
avg_latency	Average Latency	Average latency of all APIs in the gateway.	≥ 0 Unit: ms	Monitored object: dedicated API gateway Dimension: instance_id	1
max_latency	Maximum Latency	Maximum latency of all APIs in the gateway.	≥ 0 Unit: ms	Monitored object: dedicated API gateway Dimension: instance_id	1
req_count	Requests	Number of times that an API has been called.	≥ 0	Monitored object: API Dimension: api_id	1

Metric ID	Metric Name	Description	Value Range	Monitored Object and Dimension	Monitoring Period (Minute)
req_count_2xx	2xx Responses	Number of times that the API returns a 2xx response.	≥ 0	Monitored object: API Dimension: api_id	1
req_count_4xx	4xx Errors	Number of times that the API returns a 4xx error.	≥ 0	Monitored object: API Dimension: api_id	1
req_count_5xx	5xx Errors	Number of times that the API returns a 5xx error.	≥ 0	Monitored object: API Dimension: api_id	1
req_count_error	Total Errors	Total number of errors returned by the API.	≥ 0	Monitored object: API Dimension: api_id	1
avg_latency	Average Latency	Average latency of the API.	≥ 0 Unit: ms	Monitored object: API Dimension: api_id	1
max_latency	Maximum Latency	Maximum latency of the API.	≥ 0 Unit: ms	Monitored object: API Dimension: api_id	1
input_throughput	Incoming Traffic	Incoming traffic of the API.	≥ 0 Unit: Byte, KB, MB, or GB	Monitored object: API Dimension: api_id	1
output_throughput	Outgoing Traffic	Outgoing traffic of the API.	≥ 0 Unit: Byte, KB, MB, or GB	Monitored object: API Dimension: api_id	1

Metric ID	Metric Name	Description	Value Range	Monitored Object and Dimension	Monitoring Period (Minute)
node_system_load	Node System Load	Load details of a gateway node on the data plane. **1** means low water level, **2** means medium water level, and **3** means high water level.	1, 2, 3 Unit: count	Monitored object: gateway node Dimension: node_ip	1
node_cpu_usage	Node CPU Usage	CPU usage details of a gateway node on the data plane.	≥ 0 Unit: %	Monitored object: gateway node Dimension: node_ip	1
node_memory_usage	Node Memory Usage	Memory usage details of a gateway node on the data plane.	≥ 0 Unit: %	Monitored object: gateway node Dimension: node_ip	1

Dimension

Table 8-2 Monitoring dimensions

Key	Value
instance_id	Dedicated gateway
instance_id,node_ip	Dedicated gateway node
instance_id,api_id	API

8.1.2 Creating Alarm Rules

Scenario

You can create alarm rules to monitor the status of your APIs.

An alarm rule consists of a rule name, monitored objects, metrics, alarm thresholds, monitoring interval, and notification.

Prerequisites

An API has been called.

Procedure

Step 1 Go to the APIG console.

Step 2 Select a gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **API Management** > **API Groups**.

Step 4 Click a group name.

Step 5 On the **Monitoring** area of the **APIs** tab, click **More** to access the Cloud Eye console. Then create an alarm rule. For details, see section "Creating an Alarm Rule" in the *Cloud Eye User Guide*.

----End

8.1.3 Viewing Metrics

Cloud Eye monitors the status of your APIs and allows you to view their metrics.

Viewing Metrics of an API

Step 1 Go to the APIG console.

Step 2 Select a gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **API Management** > **API Groups**.

Step 4 Click a group name.

Step 5 In the left pane of the **APIs** tab, select an API.

Step 6 View metrics of the API in the **Monitoring** area.

View the call statistics of an API, including **Requests**, **Latency (ms)**, **Data Traffic (bytes)**, and **Errors**. You can also select a time range to view the data.

- Data in the last hour is updated every 2 minutes.
- Data in the last 6 hours is updated every 2 hours.
- Data in the last day is updated every 2 hours.
- Data in the last week and last month is updated every day.

Step 7 To view monitoring information about instances and instance nodes, click **More**.

NOTE

The monitoring data is retained for two days. To retain the data for a longer period, save it to an OBS bucket.

----End

Viewing Metrics of an API group

Step 1 Go to the APIG console.

Step 2 Select a gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **Monitoring & Analysis** > **API Monitoring**.

Step 4 Select the API group to be viewed and view the API call statistics, including **Requests**, **Latency (ms)**, **Data Traffic (bytes)**, and **Errors**

----End

8.2 Bandwidth Monitoring

APIG provides monitoring metrics about inbound and outbound bandwidth.

Prerequisites

Inbound and outbound access has been enabled for the target gateway. View the inbound and outbound addresses in the gateway information.

Procedure

Step 1 Go to the APIG console.

Step 2 Select a gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **Monitoring & Analysis** > **Bandwidth Monitoring**.

Step 4 Configure the monitoring information according to the following table.

Table 8-3 Monitoring information

Parameter	Description
IP Address	Inbound or outbound IP address of a gateway. View the address in the gateway information.
Time range	Select **1h**, **3h**, **12h**, **24h**, or **7d**, or click 📅 to specify a custom time range. In the upper right of each monitoring graph dynamically shows the maximum and minimum metric values in the specified time range.
Auto Refresh	If this option is enabled, data is automatically refreshed every minute.

Parameter	Description
Period	A cycle when data is aggregated to calculate the maximum, minimum, average, total, or variance value.

----End

8.3 Log Analysis

This section describes how to obtain and analyze the API calling logs of a **dedicated** gateway.

Prerequisites

APIs have been called.

Procedure

Step 1 Go to the APIG console.

Step 2 Select a gateway at the top of the navigation pane.

Step 3 In the navigation pane, choose **Monitoring & Analysis** > **Log Analysis**.

Step 4 Click **Configure Log Collection**, and change **Collect Logs** to to enable log collection.

Step 5 Specify a log group and log stream, and click **OK**. For details about log groups and log streams, see section "Log Management" in the *Log Tank Service User Guide*.

Step 6 Click **Log Fields** to view the description of each log field. Then view and analyze logs by referring to the log field descriptions.

Step 7 To export logs, see section "Log Transfer" in the *Log Tank Service User Guide*.

Fields in access logs are separated using spaces. The following table describes each log field.

Table 8-4 Log field description

No.	Field	Description
1	remote_addr	Client IP address.
2	request_id	Request ID.
3	api_id	API ID
4	user_id	Project ID provided by a requester for IAM authentication.
5	app_id	App ID provided by a requester for app authentication.

No.	Field	Description
6	time_local	Time when a request is received.
7	request_time	Request latency.
8	request_method	HTTP request method.
9	scheme	Request protocol.
10	host	Domain name.
11	router_uri	Request URI.
12	server_protocol	Request protocol.
13	status	Response status code.
14	bytes_sent	Response size in bytes, including the status line, header, and body.
15	request_length	Request length in bytes, including the start line, header, and body.
16	http_user_agent	User agent ID.
17	http_x_forwarded_for	**X-Forwarded-For** header field.
18	upstream_addr	Backend address.
19	upstream_uri	Backend URI.
20	upstream_status	Backend response code.
21	upstream_connect_time	Time taken to establish a connection with the backend.
22	upstream_header_time	Duration from the start of a connection to the first byte received from the backend.
23	upstream_response_time	Duration from the start of a connection to the last byte received from the backend.
24	region_id	Region ID.
25	all_upstream_response_time	Duration from the start of a connection to the last byte received from the backend, in seconds. When a retry occurs, the value is the total time taken.
26	errorType	API request error type. Options: ● **0**: non-throttling error ● **1**: throttling error
27	auth_type	API authentication mode.

No.	Field	Description
28	access_model1	Authentication mode 1.
29	access_model2	Authentication mode 2. Enabling two-factor authentication will use the custom authorizer ID.
30	inner_time	APIG internal processing duration, in seconds.
31	proxy_protocol_vni	VPC endpoint virtual network ID.
32	proxy_protocol_vpce_id	VPC endpoint ID.
33	proxy_protocol_addr	Client IP address.
34	body_bytes_sent	API request body size, in bytes.
35	api_name	API name.
36	app_name	Name of the app used by a requester for authentication.
37	provider_app_id	App ID of an API.
38	provider_app_name	App name of an API.
39	custom_data_log1	Custom log field 1.
40	custom_data_log2	Custom log field 2.
41	custom_data_log3	Custom log field 3.
42	custom_data_log4	Custom log field 4.
43	custom_data_log5	Custom log field 5.
44	custom_data_log6	Custom log field 6.
45	custom_data_log7	Custom log field 7.
46	custom_data_log8	Custom log field 8.
47	custom_data_log9	Custom log field 9.
48	custom_data_log10	Custom log field 10.
49	response_source	Response source. Options: ● **local**: APIG ● **remote**: backend service
50	gzip_ratio	Ratio of the original response body size to the compressed response body size.
51	upstream_scheme	Backend protocol type.

No.	Field	Description
52	group_id	Group ID.
53	apig_err_code	Gateway error code.
54	function_urn	Function URN.

----End

9 Gateway Management

9.1 Buying a Gateway

This section describes how to create a gateway. You can create APIs and use them to provide services only after a gateway is created.

Constraints

There are some limitations on creating a gateway. If you cannot create a gateway or a gateway fails to be created, check the following items:

- Gateway quota

  By default, your account can be used to create five gateways in a project. To create more dedicated gateways, submit a service ticket to increase the quota.
- Permissions

  You must be assigned both the **APIG Administrator** and **VPC Administrator** roles or assigned the **APIG FullAccess** policy to create a gateway.

  You can also be granted permissions using custom policies. For details, see **APIG Custom Policies**.
- Number of available private IP addresses in the subnet

  The basic, professional, enterprise, and platinum editions of APIG require 3, 5, 6, and 7 private IP addresses. Check that the subnet you choose has sufficient private IP addresses on the VPC console.

Network Environment

- Workload

  Gateways are deployed in VPCs (workloads). Cloud resources, such as Elastic Cloud Servers (ECSs), in the same workload can call APIs using the private IP address of the gateway deployed in the workload.

  You are advised to deploy your gateways in the same workload as your other services to facilitate network configuration and secure network access.

  NOTE

  VPCs (workloads) where gateways have been deployed cannot be changed.

- EIP

  To allow public inbound access to the APIs deployed in a gateway, create an Elastic IP (EIP) and bind it to the gateway.

  NOTE

  For APIs whose backend services are deployed on a public network, APIG automatically generates an IP address for public outbound access, and you do not need to create an Elastic IP (EIP).

- Security group

  Similar to a firewall, a security group controls access to a gateway through a specific port and transmission of communication data from the gateway to a specific destination address. For security purposes, create inbound rules for the security group to allow access only on specific ports.

  The security group bound to a gateway must meet the following requirements:

  – Inbound access: To allow the APIs in the gateway to be accessed over public networks or from other security groups, configure inbound rules for the security group to allow access on ports 80 (HTTP) and 443 (HTTPS).

  – Outbound access: If the backend service of an API is deployed on a public network or in another security group, add outbound rules for the security group to allow access to the backend service address through the API calling port.

  – If the frontend and backend services of an API are bound with the same security group and VPC as the gateway, no inbound or outbound rules are needed to allow access through the preceding ports.

Procedure

Step 1 Go to the APIG console.

NOTE

- ELB load balancing is enabled by default after gateways are purchased. Gateways with load balancing enabled do not support security groups. To disable access from specific IP addresses, use **access control policies**.
- ELB functions as a load balancer for gateways, which support cross-VPC access. Gateways with public inbound access enabled are randomly assigned an EIP and cannot use an existing EIP.

Step 2 In the navigation pane, choose **Gateways**.

Step 3 Click **Buy Gateway**. Set the gateway parameters by referring to the following table.

Table 9-1 API gateway parameters

Parameter	Description
Region	A geographic area where the gateway will be deployed. Deploy the gateway in the same region as your other services to allow all services to communicate with each other through subnets within a workload. This reduces public bandwidth costs and network latency.
AZ	A physical region where resources use independent power supplies and networks. Availability zones (AZs) are physically isolated but interconnected through an internal network. To enhance gateway availability, deploy the gateway in multiple AZs. APIG does not support gateway migration across AZs.
Gateway Name	Gateway name.
Edition	The basic, professional, enterprise, and platinum editions are available. The number of concurrent requests allowed varies depending on the gateway edition. For more information, see **Specifications** in the *API Gateway Service Overview*.
Scheduled Maintenance	Time period when the gateway can be maintained. The technical support personnel will contact you before maintenance. Select a time period with low service demands.
Public Inbound Access	Determine whether to allow the APIs created in the gateway to be called by external services using an EIP. To enable this function, assign an EIP to the dedicated gateway. **NOTE** ● APIs in the gateway can be called using independent or debugging domain names. There is a limit on the number of times that APIs in an API group can be called per day using the debugging domain name. To overcome the limitation, bind independent domain names to the API group and ensure that the domain names have already been CNAMEd to the EIP of the gateway to which the API group belongs. For example, you have an HTTPS API (path: **/apidemo**) with public access enabled. The API can be called using "**https://***{domain}***/apidemo**", where *{domain}* indicates an independent domain name bound to the group of the API. The default port is 443.

Parameter	Description
Public Outbound Access	Determine whether to allow backend services of the APIs created in the gateway to be deployed on public networks. Set a bandwidth that meets your service requirements for public outbound access. The bandwidth will be billed by hour based on the pricing of the EIP service.
Network	Select a VPC and subnet for the dedicated gateway. ● Select the created VPC and subnet from the drop-down list. ● Create a VPC and subnet by clicking **Create VPC**. For details, see section "Creating a VPC" in *Virtual Private Cloud User Guide*.
Security Group	Select a security group to control inbound and outbound access. If the backend service of an API is deployed on an external network, configure security group rules to allow access to the backend service address through the API calling port. **NOTE** If public inbound access is enabled, add inbound rules for the security group to allow access on ports 80 (HTTP) and 443 (HTTPS).
VPC Endpoint Service	Name of a VPC endpoint service to create when you buy the gateway. The gateway then can be accessed using the endpoint service. If a name is specified, the VPC endpoint service name to display on the **VPC Endpoints** tab will be in the format "*{region}*.*{Specified VPC endpoint service name}*.*{VPC endpoint service ID}*". If no name is specified, the displayed name will be in the format "*{region}***.apig.***{VPC endpoint service ID}*".
Tags	Tags classify your gateways to facilitate search, analysis, and management. If no tag is available, click **View predefined tags** or enter a tag key and value to create one. Alternatively, set tags on the Tag Management Service (TMS) console by referring to **Managing Tags**.
Description	Description about the gateway.

Step 4 Click **Next**.

Step 5 Confirm the gateway configurations. The instance is created with the status displayed on the screen.

----End

Follow-Up Operations

After the gateway is created, you can create and manage APIs in this gateway. Go to the **Gateway Information** page. It shows the gateway details, network configurations, and configuration parameters.

You can modify the gateway name, description, scheduled maintenance time window, security group, and EIP.

Before deleting a gateway, ensure that the deletion will not impact your services.

9.2 Viewing or Modifying Gateway Information

You can view and modify the configuration of your gateways on the console.

Procedure

Step 1 Go to the APIG console.

Step 2 In the navigation pane, choose **Gateways**.

Step 3 Click **Access Console** or the name of the target gateway.

Step 4 On the **Gateway Information** tab, view or modify the configuration of the gateway.

Table 9-2 Gateway information

Modifiable Parameter	Description
Basic Information	Basic information about the gateway, including the name, ID, edition, AZ, description, enterprise project, and maintenance time window. ● Modify the basic information as required. ● To copy the gateway ID, click ⧉ next to the ID.
Billing Mode	Billing mode of the gateway and the time when the gateway is created.
Network	● VPC VPC associated with the gateway. Click the VPC name to view the configuration. ● Subnet Subnet associated with the gateway. Click the subnet name to view the configuration. ● Security Group Security group associated with the gateway. Click the security group name to view the configuration or click ✎ to bind a new one.

Modifiable Parameter	Description
Inbound Access	● VPC Access Address ● EIP – To bind an EIP to the gateway, click **Enable**. – To copy the bound EIP, click ❐. – Modify the bandwidth as required. The bandwidth is billed by hour based on the rate of the EIP service. – To unbind the EIP from the gateway, click **Unbind EIP**.
Outbound Access	Determine whether to allow backend services of the APIs created in the gateway to be deployed on public networks. You can enable or disable outbound access at any time. After enabling outbound access, you can click **View Metrics** to view the monitoring data, or modify the bandwidth as required.
Routes	Configure a private network segment that needs to communicate with the gateway. After a gateway is created, it can communicate with the VPC subnet specified during gateway creation by default. Configure routes at your premises if the subnet of your data center is within the following three segments: 10.0.0.0/8-24, 172.16.0.0/12-24, and 192.168.0.0/16-24.

----End

9.3 Configuring Parameters

This section describes how to configure common parameters for a gateway to adjust component functions.

Constraint

Modifying gateway configuration parameters will interrupt services. Do this during off-peak hours or when no service is running.

Procedure

Step 1 Go to the APIG console.

Step 2 In the navigation pane, choose **Gateways**.

Step 3 Click **Access Console** or the name of the target gateway.

Step 4 Click the **Parameters** tab, and click **Modify** in the row that contains the target parameter. The configuration parameters vary depending on the gateway edition.

Table 9-3 Configuration parameters

Parameter	Description
ratelimit_api_limits	Default request throttling value applied to all APIs. Default: 200 calls/second. The total number of times an API can be called is determined by this parameter only if no request throttling policy is bound to the API. The **Max. API Requests** of a request throttling policy cannot exceed the value of this parameter.
request_body_size	Maximum size of the body that can be carried in an API request. The default value is 12 MB. The value ranges from 1 MB to 9,536 MB.
backend_timeout	Backend response timeout. Default: 60,000 ms. Range: 1–600,000 ms.
app_token	Determine whether to enable app_token authentication. Default: disabled. If you enable this function, an access_token can be added to the API request for authentication. ● **app_token_expire_time**: validity period of an access_token. A new access_token must be obtained before the original access_token expires. ● **refresh_token_expire_time**: the validity period of a refresh_token. A refresh_token is used to obtain a new access_token. ● **app_token_uri**: the URI used to obtain an access_token. ● **app_token_key**: the encryption key of an access token.
app_basic	Determine whether to enable app_basic authentication. Default: disabled. After this option is enabled, users can add the header parameter **Authorization** and set the parameter value to "Basic + base64 (*appkey* + : + *appsecret*)", in which *appkey* and *appsecret* are the key and secret of a credential.
app_secret	Determine whether to enable app_secret authentication. Default: disabled. If you enable this function, the **X-HW-ID** and **X-HW-AppKey** parameters can be added to the API request to carry the key and secret of a credential for authentication.
app_route	Determine whether to support IP address–based API access. Default: disabled. If you enable this function, APIs in any group except DEFAULT can be called using IP addresses.

Parameter	Description
backend_client_certificate	Determine whether to enable backend two-way authentication. Default: disabled. If you enable this function, you can configure two-way authentication for a backend when creating an API.
ssl_ciphers	Supported HTTPS cipher suites. By default, all cipher suites are supported. Select cipher suites after you bind independent domain names to an API group.
real_ip_from_xff	Determine whether to use the IP addresses in the **X-Forwarded-For** header for access control and request throttling. By default, the IP addresses in this header are not used. **xff_index**: Sequence number of an IP address in the **X-Forwarded-For** header. The value can be positive, negative, or 0. ● If the value is 0 or positive, the IP address of the corresponding index in the **X-Forwarded-For** header will be obtained. ● If the value is negative, the IP address of the indicated reverse sequence in the **X-Forwarded-For** header will be obtained. For example, assume that the **X-Forwarded-For** header of a request received by API gateway contains three IP addresses: IP1, IP2, and IP3. If the value of **xff_index** is 0, IP1 is obtained. If the value is 1, IP2 is obtained. If the value is –1, IP3 is obtained. If the value is –2, IP2 is obtained.
vpc_name_modifiable	Determine whether load balance channel names can be modified. By default, the names can be modified. **NOTICE** If this option is enabled, load balance channels of the current gateway cannot be managed using project-level load balance channel management APIs.

Parameter	Description
custom_auth_header	Determine whether to support custom authentication headers. By default, custom authentication headers are not supported. If you enable this parameter, the initial values of **app_auth_header** and **backend_sign_header** are empty, same as when the parameter is disabled. If you set the **Current Value** of **app_auth_header**, the parameter with the same name as this value carries the app authentication information in the request header for APIs that use app authentication. If you set the **Current Value** of **backend_sign_header**, the parameter with the same name as this value carries the signature information in the backend request header for APIs bound with an HMAC or Basic Auth signature key policy. **NOTICE** Configuring this parameter will affect all APIs that use app authentication or are bound with an HMAC or Basic Auth signature key policy in the gateway.
gzip	Determine whether to compress responses using gzip to reduce public network traffic. By default, responses are not compressed. The configuration will take effect in 1 minute. After enabling this parameter, set the compression level parameter **comp_level**. The greater the value is, the better responses are compressed. Default: **6**. **NOTICE** ● Use gzip to compress response body larger than 1 KB. ● gzip supports the following file types: text/xml, text/plain, text/css, application/javascript, application/x-javascript, application/rss+xml, text/javascript, image/tiff, image/svg+xml, application/json, and application/xml. ● After enabling gzip compression, you must add request header **Accept-Encoding: gzip**. ● The gzip configuration can be modified 1 minute after being completed.

Parameter	Description
custom_log	Whether to enable custom logs. Default: disabled. Once enabled, values of specified parameters will be printed in specified locations of **calling logs** for all APIs in the gateway. After this function is enabled, click **Modify**, and then click **Add** to add the parameters to print in calling logs. **NOTICE** ● Custom logs print only the requests initiated from clients and do not print the constants and system parameters defined in APIG. ● Custom logs can have a maximum of 10 fields, with a total size of not more than 2 KB. ● Some special characters in parameter values will be encoded. For example, the plus sign (+) will be encoded as a space, double quotation marks (") encoded as **\x22**, and a backslash (\) encoded as **\x5C**.
sse_strategy	Whether to enable Server-Sent Events (SSE) transmission. It is disabled by default. Once enabled, the responses of APIs are output in streaming mode for character-based rendering. **NOTICE** The sse_strategy configuration can be modified 1 minute after being completed.
vpc_name_modifiable	Whether to enable name modification of the load balance channel. If the load balance channel name is modifiable, the load balance channel of the current gateway cannot be operated through the project-level load balance channel management API.
request_custom_confi g	Configure client request parameters. ● **HTTP/2**: Enabled by default. For details, see **HTTP 2.0**. ● **request_body_timeout**: Timeout for client request body. Default: 8s. Modify this parameter if the network condition is poor or the request body is too large. **NOTICE** The client request configuration can be modified 1 minute after being completed.
api_uri_no_escape	Determine whether to escape the path in the API URL. It is disabled by default, indicating that the path in the URL is escaped. For details about the function of not escaping paths after **api_uri_no_escape** is enabled, see **Table 9-4**.

Table 9-4 Functions affected if path is not escaped

Function	Description	API Frontend Definition Path	Path for Sending a Request	Disabling api_uri_no_escape	Enabling api_uri_no_escape
API definition	Path for APIG to match routes.	/{path}	/aa %2Faa	/aa/aa	/aa%2Faa
Parameter orchestration	Path used by backend service parameters.	-	-	/aa/aa	/aa%2Faa
HTTP-to-HTTPS redirection	Path used for redirection.	-	-	/aa/aa	/aa%2Faa
Backend policies	The policy condition is the path of the request input parameter.	-	-	/aa/aa	/aa%2Faa
Third-party authentication policy	Path transferred to the third-party system after the API is bound to a third-party authentication policy.	-	-	/aa/aa	/aa%2Faa
Kafka log push policy	Request path used after the Kafka log push policy is bound to the API.	-	-	/aa/aa	/aa%2Faa
Load balance channels	Path used by APIG to forward requests when the load balance channel uses the URI hash.	-	-	/aa/aa	/aa%2Faa

Function	Description	API Frontend Definition Path	Path for Sending a Request	Disabling api_uri_no_escape	Enabling api_uri_no_escape
FunctionGraph backends	Request path sent to a function when the backend type of the API is FunctionGraph.	-	-	/aa/aa	/aa%2Faa
Custom authentication	Path of the request sent to the function when the API authentication mode is set to **Custom**.	-	-	/aa/aa	/aa%2Faa

----End

9.4 Managing Tags

Tags classify your gateways to facilitate search, analysis, and management.

You can search, analyze, and manage your resources on the Tag Management Service (TMS) console.

Procedure

Step 1 Go to the APIG console.

Step 2 In the navigation pane, choose **Gateways**.

Step 3 Click **Access Console** or the name of the target gateway.

Step 4 On the **Tags** tab, click **Add Tag**.

A tag consists of a key and value. The value can be empty.

Step 5 Click **OK**.

----End

Related Operations

Use the tags to filter, view, analyze, and manage your gateway resources on the TMS console.

Step 1 Hover over ☰ on the left to expand the service list, and enter **TMS**.

Step 2 On the TMS console, enter the required information to filter gateway resources.

1. **Region**: Select the region where the gateway is.
2. **Resource Type**: Select **APIG**.
3. **Resource Tag**: Select a tag key.

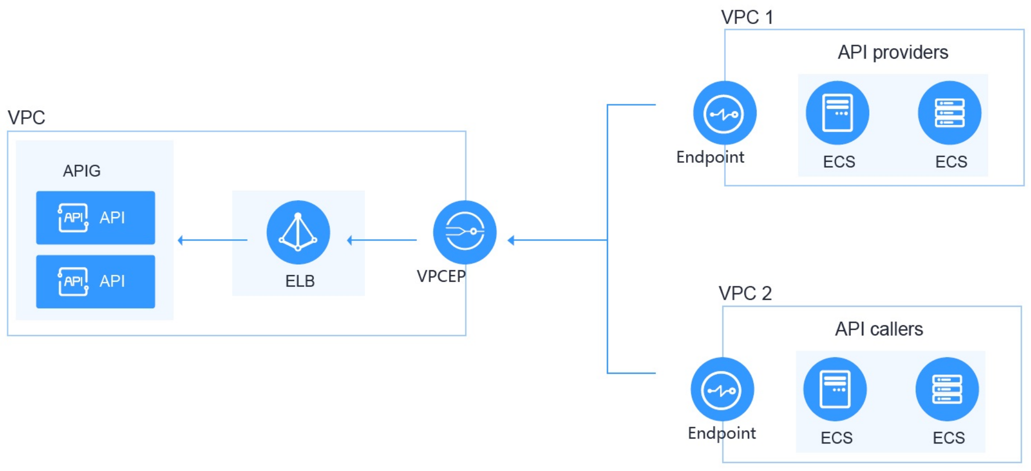
----End

9.5 Managing VPC Endpoints

VPC endpoints are secure and private channels for connecting VPCs to VPC endpoint services.

APIs can be exposed and accessed across VPCs in the same region of the same cloud.

Figure 9-1 Cross-VPC access in the same region



Procedure

Step 1 Go to the APIG console.

Step 2 In the navigation pane, choose **Gateways**.

Step 3 Click **Access Console** or the name of the target gateway.

Step 4 Click **VPC Endpoints** to view details. For details, see section "VPC Endpoints" in the *VPC Endpoint User Guide*.

Table 9-5 VPC endpoint information

Parameter	Description
VPC Endpoint Service	Display name of the VPC endpoint service in the format "*{region}.{VPC endpoint service name}.{VPC endpoint service ID}*". You can set the VPC endpoint service name when **buying a gateway** or later on the **VPC Endpoints** tab of the gateway.

Parameter	Description
Connections	VPC endpoints connected to the gateway. If you need a new VPC endpoint, click **Create VPC Endpoint**. ● **VPC Endpoint ID**: ID of a VPC endpoint. ● **Packet ID**: identifier of the VPC endpoint ID. ● **Status**: status of the VPC endpoint. For details about VPC endpoint statuses, see section "What Are Statuses of VPC Endpoint Services and VPC Endpoints?" in the *VPC Endpoint User Guide*. ● **Owner**: of the VPC endpoint creator. ● **Created**: time when the VPC endpoint is created. ● **Operation**: whether to allow the VPC endpoint to connect to the VPC endpoint service. Accept or reject connection from the VPC endpoint to the VPC endpoint service. **NOTICE** Once you reject the connection, services that run using the connection may be affected. Exercise caution.
Permissions	Specify accounts allowed to access using the VPC endpoints by adding the account IDs to the whitelist. Click **Add Account** and enter an account ID. To obtain the account ID, see "Obtaining an Account Name and Account ID" in the *API Gateway API Reference*. ● **Account ID**: ID of an account allowed to access using the VPC endpoints. ● **Created**: time when the whitelist is created. ● **Operation**: Manage access of the account from VPC endpoints. To forbid access of the account, remove it from the whitelist.

----End

9.6 Modifying Specifications

If the specifications of a gateway cannot meet your service requirements, upgrade the specifications.

NOTICE

- During the specification change, the persistent connection is intermittently disconnected and needs to be re-established. You are advised to change the specification during off-peak hours.
- Specifications can be upgraded but cannot be downgraded.
- Changing the gateway edition will also change the private network access IP addresses. Modify your firewall or whitelist configuration if necessary for service continuity. Do not perform any other operations on the gateway.

Procedure

Step 1 Go to the APIG console.

Step 2 In the navigation pane, choose **Gateways**.

Step 3 Choose **More** > **Modify Specifications** on the right of the target gateway.

Step 4 Select an edition and click **Next**. For details about the gateway parameters, see **Table 9-3**.

Step 5 Confirm the configuration, read and confirm your acceptance of the service agreement, and click **Pay Now**. The upgrade takes 15 to 30 minutes to complete.

NOTE

- For pay-per-use gateways, pay for what you use without needing to pay for any extra fees.

----End

10 SDKs

APIG supports API authentication based on IAM, apps, and custom authorizers. You can also choose not to authenticate API requests. For details about the differences between the four modes and how to select one, see **Calling APIs**. This section describes how to download SDKs and view related instructions.

Scenario

SDKs are used when you call APIs through app authentication. Download SDKs and related documentation and then call APIs by following the instructions in the documentation.

Procedure

Step 1 Go to the APIG console.

Step 2 In the navigation pane, choose **Help Center**.

Step 3 Click **Using SDKs**.

Step 4 Click **Download SDK** next to the desired language. An SDK contains SDK code and sample code. SDKs vary depending on the language.

To view the support guide, click **SDK Documentation**.

----End

11 Published API Calling

11.1 Calling APIs

You can call APIs opened by others in APIG.

Usage Guidelines

- An API can be accessed 1000 times by using the debugging domain name allocated when the API's group is created.
- If the **CA** parameter is displayed in the **Create SSL Certificate** dialog box on the **API Management** > **API Policies** > **SSL Certificates** page of the APIG console, pay attention to the following restrictions when calling APIs:
  - When calling an API with HTTP/1.0, do not use **Transfer-Encoding** in the request header.
  - Do not use the CONNECT method.
  - Do not use both **Content-Length** and **Transfer-Encoding** in the request header.
  - Do not use spaces or control characters in the request line.
  - Do not use spaces or control characters in the header name.
  - Do not use spaces or control characters in the **Host** request header.
  - Dot not use multiple **Host** parameters in the request header.

Prerequisites

Before calling an API, ensure that the network of your service system can communicate with the API access domain name or address.

- If the service system and gateway are in the same VPC, the API can be directly accessed.
- If the service system and gateway are in different VPCs of a region, connect them using a peering connection. For details, see section "VPC Peering Connection" in the *Virtual Private Cloud User Guide*.
- If the service system and gateway are in different VPCs of different regions, create a cloud connection and load the two VPCs to connect them. For details,

see section "Connecting VPCs in Different Regions" in the *Cloud Connect Getting Started*.

- If the service system and gateway are connected over the public network, ensure that the gateway has been bound with an EIP.

Obtaining API Calling Information

Obtain API calling information from the API provider before you call an API.

- Obtain API request information

  On the APIG console, choose **API Management** > **APIs**. On the **APIs** page, obtain the domain name, request method, and request path of the desired API. Click the API name to go to the **APIs** tab page, and obtain the basic information in the **Frontend Configuration** and **Backend Configuration** areas.

- Obtain API authentication information

  Obtain the request authentication information according to the API's authentication mode.

Authentication Mode	Authentication Information
App (signature)	Obtain the key and secret of a credential authorized for the API from the API provider, as well as the signing SDK.
App (simple authentication)	Obtain the AppCode of a credential authorized for the API from the API provider.
App (two-factor)	Obtain the information required for both app and custom authentication.
App (app_secret)	Obtain the key and secret of a credential authorized for the API from the API provider.
App (app_basic)	Obtain the key and secret of a credential authorized for the API from the API provider.
IAM (token)	Obtain the username and password for the cloud platform.
IAM (AK/SK)	Obtain the AK/SK of an account for the cloud platform and the signing SDK.
IAM (two-factor)	Obtain the information required for both IAM and custom authentication
Custom	Obtain the custom authentication information to carry in request parameters from the API provider.
None	No authentication information required.
Third-party authorizer (API policy)	Obtain third-party authorizer information to carry in request parameters from the API provider.

- Credential key and secret

  On the APIG console, choose **API Management** > **Credentials**. Click the name of a credential authorized for the target API, and obtain the key and secret on the credential details page.

- Signing SDK

  On the APIG console, choose **Help Center** > **Using SDKs**, and download the SDK of the desired language.

- AppCode

  On the APIG console, choose **API Management** > **Credentials**. Click the name of a credential authorized for the target API, and obtain an AppCode in the **AppCodes** area of the credential details page.

Calling an API

NOTE

This section describes only the configuration of the request path and authentication parameters. For other parameters, such as timeout and SSL, configure them as required. To avoid service loss due to incorrect parameters, configure them by referring to the industry standards.

1. Construct an API request. Example:

```
POST https://{Address}/{Path}?{Query}
{Header}

{
  {Body}
}
```

   - **POST**: request method. Replace it with the request method obtained in **Obtaining API Calling Information**.
   - *{Address}*: request address. Replace it with the domain name obtained in **Obtaining API Calling Information**.

Scenario	Request Parameter Configuration
Calling an API with a domain name	Call an API using the debugging domain name allocated to the API group or a domain name bound to the group. No additional configuration is required.
Calling an API in the **DEFAULT** group with an IP address	Call an API in the **DEFAULT** group with an IP address. No additional configuration is required.

Scenario	Request Parameter Configuration
Calling an API in a custom group with an IP address	● To use an IP address to call an API that uses app authentication in a non-DEFAULT group, 1. Set **configuration parameters** **app_route** and **app_secret** of the gateway to **On**. After **app_route** is enabled, a credential cannot be authorized to APIs that use the same request path and method. 2. Add header parameters **X-HW-ID** and **X-HW-APPKEY** and set them to the key and secret of a credential authorized for the API. **NOTICE** When calling an API through simple authentication (App authentication), you only need to add the header parameters **X-Apig-AppCode** and **host** to the request. ● To use an IP address to call an API that does not use app authentication in a non-DEFAULT group, add the header parameter **host**.

- *{Path}*: request path. Replace it with the request path obtained in **Obtaining API Calling Information**.
- *{Query}*: (optional) query string in format "*Parameter_name*=*Parameter_value*", for example, **limit=10**. Separate multiple query strings with ampersands (&). For details, see the request parameters obtained in **Obtaining API Calling Information**.
- *{Header}*: request header parameter in format "*Parameter_name*:*Parameter_value*", for example, **Content-Type:application/json**. For details, see the request parameters obtained in **Obtaining API Calling Information**.
- *{Body}*: request body in JSON format. For details, see the request body description obtained in **Obtaining API Calling Information**.

2. Add authentication information to the API request.

Authentication Mode	Request Parameter Configuration
App (signature)	Use the obtained SDK to sign the API request. For details, see section "Calling APIs Through App Authentication" in the *API Gateway Developer Guide*.
App (simple authentication)	Add the header parameter **X-Apig-AppCode** and set the parameter value to the AppCode obtained in **Obtaining API Calling Information**. For details, see **Getting Started**.

Authentication Mode	Request Parameter Configuration
App (app_secret)	● Set the **app_secret** parameter to **on** on the **Parameters** tab of a gateway to enable app_secret authentication. ● Add the header parameter **X-HW-ID** and set the parameter value to the key obtained in **Obtaining API Calling Information**. ● Add the header parameter **X-HW-AppKey** and set the parameter value to the secret obtained in **Obtaining API Calling Information**.
App (app_basic)	● To enable app_basic authentication, ensure that the **app_basic** parameter has been set to **on** on the **Parameters** tab of the gateway. ● Add the header parameter **Authorization** to the API request. The value is **"Basic "+base64(appkey+":"+appsecret)**. **appkey** and **appsecret** are the key and secret obtained in **Obtaining API Calling Information**.
App (two-factor)	Add the information required for both app and custom authentication to the API request.
IAM (token)	Obtain a token from the cloud platform and add the header parameter **X-Auth-Token** with the token as the value. For details, see section "Token Authentication" in the *API Gateway Developer Guide*.
IAM (AK/SK)	Use the obtained SDK to sign the API request. For details, see section "AK/SK Authentication" in the *API Gateway Developer Guide*.
IAM (two-factor)	Add the information for both IAM and custom authentication to the API request.
Custom	Add the information required for custom authentication to the API request.
None	No authentication information required.
Third-party authorizer (API policy)	Obtain third-party authorizer information to carry in request parameters from the API provider.

11.2 Response Headers

The following table describes the response headers that APIG adds to the response returned when an API is called.

X-Apig-Mode: debug indicates API debugging information.

Response Header	Description	Remarks
X-Request-Id	Request ID.	Returned for all valid requests.
X-Apig-Latency	Duration from the time when APIG receives a request to the time when the backend returns a message header.	Returned only when the request header contains **X-Apig-Mode: debug**.
X-Apig-Upstream-Latency	Duration from the time when APIG sends a request to the backend to the time when the backend returns a message header.	Returned only when the request header contains **X-Apig-Mode: debug** and the backend type is not Mock.
X-Apig-RateLimit-api	API request limit information. Example: **remain:9,limit:10,time:10 second**.	Returned only when the request header contains **X-Apig-Mode: debug** and a limit has been configured for the number of times the API can be called.
X-Apig-RateLimit-user	User request limit information. Example: **remain:9,limit:10,time:10 second**.	Returned only when the request header contains **X-Apig-Mode: debug** and a limit has been configured for the number of times the API can be called by a user.
X-Apig-RateLimit-app	Credential request limit information. Example: **remain:9,limit:10,time:10 second**.	Returned only when the request header contains **X-Apig-Mode: debug** and a limit has been configured for the number of times the API can be called by a credential.
X-Apig-RateLimit-ip	IP address request limit information. Example: **remain:9,limit:10,time:10 second**.	Returned only when the request header contains **X-Apig-Mode: debug** and a limit has been configured for the number of times the API can be called by an IP address.
X-Apig-RateLimit-api-allenv	Default API request limit information. Example: **remain:199,limit:200,time:1 second**.	Returned only when the request header contains **X-Apig-Mode: debug**.

11.3 Error Codes

The following table lists the error codes that you may encounter when calling APIs. If an error code starting with **APIGW** is returned after you call an API, rectify the fault by referring to the instructions provided in section "Error Codes" in the *API Request Signing Guide*.

NOTE

- For details about the error codes that may occur when you manage APIs, see section "Error Codes" in the *API Gateway API Reference*.
- If an error occurs when you use APIG, find the error message and description in the following table according to the error code, for example, APIG.0101. The error messages are subject to change without prior notice.

Table 11-1 Error codes

Error Code	Error Message	HTTP Status Code	Description	Solution
APIG.0101	The API does not exist or has not been published in the environment.	404	The API does not exist or has not been published in the environment.	Check whether the domain name, method, and path are consistent with those of the created API. Check whether the API has been published. If it has been published in a non-production environment, check whether the **X-Stage** header in the request is the environment name. Check whether the domain name used to call the API has been bound to the group to which the API belongs.

Error Code	Error Message	HTTP Status Code	Description	Solution
APIG.0101	The API does not exist.	404	The API request method does not exist.	Check whether the API request method is the same as the method defined by the API.
APIG.0103	The backend does not exist.	500	The backend service was not found.	Contact technical support.
APIG.0104	The plug-ins do not exist.	500	No plug-in configurations were found.	Contact technical support.
APIG.0105	The backend configurations do not exist.	500	No backend configurations were found.	Contact technical support.
APIG.0106	Orchestration error.	400	An orchestration error occurred.	Check whether the frontend and backend parameters of the API are correct.
APIG.0107	The custom lua script encountered an unexpected error	500	An unknown error occurred in the Lua script.	Contact technical support.
APIG.0201	API request error.	400	Invalid request parameters.	Set valid request parameters.
APIG.0201	Request entity too large.	413	The request body exceeds 12 MB.	Reduce the size of the request body.
APIG.0201	Request URI too large.	414	The request URI exceeds 32 KB.	Reduce the size of the request URI.
APIG.0201	Request headers too large.	494	The request headers are too large because one of them exceeds 32 KB or the total length exceeds 128 KB.	Reduce the size of the request headers.

Error Code	Error Message	HTTP Status Code	Description	Solution
APIG.0201	Backend unavailable.	502	The backend service is unavailable.	Check whether the backend address configured for the API is accessible.
APIG.0201	Backend timeout.	504	The backend service has timed out.	Increase the timeout duration of the backend service or shorten the processing time.
APIG.0201	An unexpected error occurred	500	An internal error occurred.	Contact technical support.
APIG.0202	Backend unavailable	502	The backend is unavailable.	Check whether the backend request protocol configured for the API is the same as the request protocol used by the backend service.
APIG.0203	Backend timeout	504	The backend service has timed out.	Increase the timeout duration of the backend service or shorten the processing time.
APIG.0204	SSL protocol is not supported: TLSv1.1	400	The SSL protocol version is not supported.	Use a supported SSL protocol version.
APIG.0205	Verify client certificate failed	400	Failed to verify the client certificate.	Check whether the client certificate is correct.
APIG.0301	Incorrect IAM authentication information.	401	The IAM authentication details are incorrect.	Check whether the token is correct.

Error Code	Error Message	HTTP Status Code	Description	Solution
APIG.0302	The IAM user is not authorized to access the API.	403	The IAM user is not allowed to access the API.	Check whether the user is controlled by a blacklist or whitelist.
APIG.0303	Incorrect app authentication information.	401	The app authentication details are incorrect.	Check whether the request method, path, query strings, and request body are consistent with those used for signing; check whether the date and time on the client are correct; and check whether the signing code is correct by referring to section "Calling APIs Through App Authentication" of the *Developer Guide*.
APIG.0304	The app is not authorized to access the API.	403	The app is not allowed to access the API.	Check whether the app has been authorized to access the API.
APIG.0305	Incorrect authentication information.	401	The authentication information is incorrect.	Check whether the authentication information is correct.
APIG.0306	API access denied.	403	Access to the API is not allowed.	Check whether you have been authorized to access the API.
APIG.0307	The token must be updated.	401	The token needs to be updated.	Obtain a new token from IAM.

Error Code	Error Message	HTTP Status Code	Description	Solution
APIG.0308	The throttling threshold has been reached.	429	The throttling threshold has been reached.	Try again after the throttling resumes. If the number of debugging domain requests per day is reached, bind an independent domain name to the service to which the API belongs.
APIG.0310	The project is unavailable.	403	The project is currently unavailable.	Select another project and try again.
APIG.0311	Incorrect debugging authentication information.	401	The debugging authentication details are incorrect.	Contact technical support.
APIG.0312	Incorrect third-party authentication information,auth fail	401	The authentication failed because the third-party authentication information is incorrect.	Check whether the identity information is correct.
APIG.0313	Incorrect third-party authentication information,identities error	401	The identity included in the third-party authentication information is incorrect.	Check whether the identity information is consistent with the identity source in the third-party authentication plug-in.

Error Code	Error Message	HTTP Status Code	Description	Solution
APIG.0314	Incorrect third-party authentication information,access deny	403	Access denied because the third-party authentication information is incorrect.	Contact technical support to check whether the request is a service request. If yes, increase the brute force threshold of the third-party authentication plug-in.
APIG.0401	Unknown client IP address.	403	The client IP address cannot be identified.	Contact technical support.
APIG.0402	The IP address is not authorized to access the API.	403	The IP address is not allowed to access the API.	Check whether the IP address is controlled by a blacklist or whitelist.
APIG.0404	Access to the backend IP address has been denied.	403	The backend IP address cannot be accessed.	Check whether the backend IP address or the IP address corresponding to the backend domain name is accessible.
APIG.0405	The app is not accessed from a trusted IP address.	403	The application is not accessed from a trusted IP address.	Check whether the source IP address is allowed or denied in the access control policy.
APIG.0501	The app quota has been used up.	405	The app quota has been reached.	Increase the app quota.
APIG.0502	The app has been frozen.	405	The app has been frozen.	Check whether your account balance is sufficient.
APIG.0601	Internal server error.	500	An internal error occurred.	Contact technical support.

Error Code	Error Message	HTTP Status Code	Description	Solution
APIG.0602	Bad request.	400	Invalid request.	Check whether the request is valid.
APIG.0605	Domain name resolution failed.	500	Domain name resolution failed.	Check whether the domain name is correct and has been bound to a correct backend address.
APIG.0606	Failed to load the API configurations.	500	API configurations could not be loaded.	Contact technical support.
APIG.0607	The following protocol is supported: {xxx}	400	The protocol is not supported. Only *xxx* is supported. *xxx* is subject to the actual value in the response.	Use HTTP or HTTPS to access the API.
APIG.0608	Failed to obtain the admin token.	500	The administrator account details cannot be obtained.	Contact technical support.
APIG.0609	The VPC backend does not exist.	500	The workload backend service cannot be found.	Contact technical support.
APIG.0610	No backend available.	502	No backend services are available.	Check whether all backend services are available. For example, check whether the API calling information is consistent with the actual configuration.

Error Code	Error Message	HTTP Status Code	Description	Solution
APIG.0611	The backend port does not exist.	500	The backend port was not found.	Contact technical support.
APIG.0612	An API cannot call itself.	500	An API cannot call itself.	Modify the backend configurations, and ensure that the number of layers the API is recursively called does not exceed 10.
APIG.0613	The IAM service is currently unavailable.	503	IAM is currently unavailable.	Contact technical support.
APIG.0615	Incorrect third-party authentication VPC information	500	Failed to obtain the load balance channel nodes for third-party authentication.	Check whether the load balance channel for third-party authentication is correctly configured.
APIG.0616	Incorrect third-party authentication request information	500	Failed to connect to the third-party authentication service.	Check whether the third-party authentication service is normal.
APIG.0617	Incorrect third-party authentication response information	500	Failed to obtain response from the third-party authentication service.	Check whether the third-party authentication service is normal.
APIG.0705	Backend signature calculation failed.	500	Backend signature calculation failed.	Contact technical support.
APIG.0802	The IAM user is forbidden in the currently selected region	403	The IAM user is disabled in the current region.	Contact technical support.
APIG.2102	PublicKey is null	400	The signature key is not found.	Contact technical support.

Error Code	Error Message	HTTP Status Code	Description	Solution
APIG.2201	Appkey or SecretKey is invalid	400	Invalid AppKey or SecretKey.	Check whether the AppKey and SecretKey in the request are correct.
APIG.2202	Refresh token is invalid	400	Invalid refresh token.	Check whether the refresh token is correct.
APIG.2203	Access token is invalid	400	Invalid access token.	Check whether the access token is correct.
APIG.2204	ContentType invalid	400	Invalid ContentType.	Check whether the ContentType is correct.
APIG.2205	Auth parameter invalid	400	Invalid authentication parameter.	Check whether the authentication parameters are correct.
APIG.2206	Auth method invalid	400	Invalid authentication mode.	Check whether the authentication mode is correct.
APIG.2208	The length of through_data is out of range	400	The length of through_data is out of range.	The maximum length of through_data is 300. Adjust through_data based on the actual situation.
APIG.2209	The value of grant_type is not in enum List	400	The value of grant_type is invalid.	The value of grant_type can only be **client_credentials** or **refresh_token**. Change it based on the actual situation.
APIG.2210	Lack of grant_type	400	The authorization type is missing.	Add grant_type.
APIG.2211	Lack of client_id	400	The client ID is missing.	Add a client ID.

Error Code	Error Message	HTTP Status Code	Description	Solution
APIG.2212	Lack of client_secret	400	The client secret is missing.	Add a client secret.
APIG.2213	Lack of refresh_token	400	The refresh token is missing.	Contact technical support.
APIG.1001	Refresh token is expired	401	The refresh token has expired.	Obtain another refresh token.
APIG.1002	Access token is expired	401	The access token has expired.	Obtain another access token.
APIG.1003	App not match refresh token	401	The app does not match the refresh token.	Check whether the client_id is correct.
APIG.1004	App not exist	401	The app does not exist.	Check whether the access token is correct.
APIG.1009	AppKey or AppSecret is invalid	400	The AppKey or AppSecret is invalid.	Check whether the AppKey or AppSecret in the request is correct.

12 Permissions Management

12.1 Creating a User and Granting APIG Permissions

This topic describes how to use Identity and Access Management (IAM) to implement fine-grained permissions control for your APIG resources. With IAM, you can:

- Create IAM users for employees based on your enterprise's organizational structure. Each IAM user will have their own security credentials for accessing APIG resources.
- Grant only the permissions required for users to perform a specific task.
- Entrust another account or cloud service to perform O&M on your APIG resources.

If your account does not require individual IAM users, skip this chapter.

This section describes the procedure for granting permissions (see **Figure 12-1**).

Prerequisites

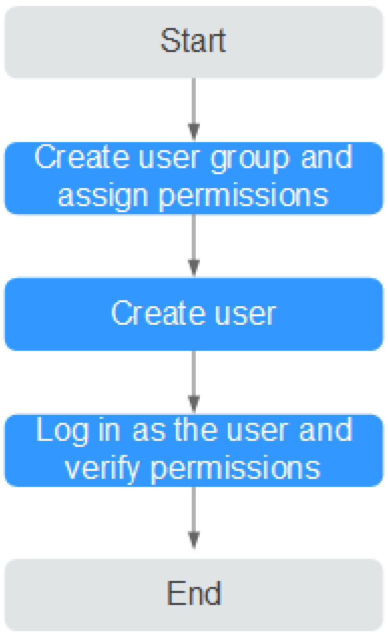
Learn about the permissions (see **Table 12-1**) supported by APIG and choose policies or roles according to your requirements.

Table 12-1 System-defined roles and policies supported by APIG

Role/ Policy Name	Description	Type	Dependency
APIG Administrator	Administrator permissions for APIG. Users granted these permissions can use all functions of API gateways.	System-defined role	If a user needs to create, delete, or change resources of other services, the user must also be granted administrator permissions of the corresponding services in the same project.
APIG FullAccess	Full permissions for APIG. Users granted these permissions can use all functions of gateways.	System-defined policy	None
APIG ReadOnly Access	Read-only permissions for APIG. Users granted these permissions can only view gateways.	System-defined policy	None

Process Flow

Figure 12-1 Process for granting APIG permissions


Start
Create user group and assign permissions
Create user
Log in as the user and verify permissions
End

1. Create a user group and assign permissions.

Create a user group on the IAM console, and attach the **APIG Administrator** role or the **APIG FullAccess** policy to the group.

2. Create an IAM user.

   Create a user on the IAM console and add the user to the group created in **1**.

3. Log in and verify permissions.

   Log in to the APIG console as the created user, and verify that the user has administrator permissions for APIG.

12.2 APIG Custom Policies

Custom policies can be created to supplement the system-defined policies of APIG. For the actions that can be added to custom policies, see section "Permissions Policies and Supported Actions" in the *API Gateway API Reference*.

You can create custom policies using one of the following methods:

- Visual editor: Select cloud services, actions, resources, and request conditions. This does not require knowledge of policy syntax.
- JSON: Edit JSON policies from scratch or based on an existing policy.

For operation details, see section "Creating a Custom Policy" in the *Identity and Access Management User Guide*. The following section contains examples of common APIG custom policies.

Example Custom Policies

- Example 1: Allow users to create and debug APIs

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "
                    apig:apis:create
                    apig:apis:debug
                "
            ]
        }
    ]
}
```

- Example 2: Deny API group creation

  A policy with only "Deny" permissions must be used in conjunction with other policies to take effect. If the permissions assigned to a user contain both "Allow" and "Deny", the "Deny" permissions take precedence over the "Allow" permissions.

  The following method can be used if you need to assign permissions of the **APIG FullAccess** policy to a user but you want to prevent the user from creating API groups. Create a custom policy for denying API group creation, and attach both policies to the group to which the user belongs. Then, the user can perform all operations on API gateways except creating API groups. The following is an example of a deny policy:

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "
                    apig:apis:create
                    apig:apis:debug
                 "
            ]
        }
    ]
}
```

13 Auditing

13.1 APIG Operations Recorded by CTS

Enabling CTS

If you want to collect, record, or query operation logs for APIG in common scenarios such as security analysis, audit, and problem locating, enable Cloud Trace Service (CTS). For details, see section "Enabling CTS" in the *Cloud Trace Service User Guide*.

CTS provides the following functions:

- Recording audit logs
- Querying audit logs
- Dumping audit logs
- Encrypting trace files
- Enabling notifications of key operations

Viewing Key Operations

With CTS, you can record operations associated with APIG for future query, audit, and backtracking.

Table 13-1 APIG operations recorded by CTS

Operation	Resource Type	Trace Name
Creating an API group	ApiGroup	createApiGroup
Modifying an API Group	ApiGroup	updateApiGroup
Deleting an API group	ApiGroup	deleteApiGroup
Verifying an API group name	Swagger	CheckApiGroups

Operation	Resource Type	Trace Name
Creating an environment	Environment	createEnvironment
Modifying an environment	Environment	updateEnvironment
Deleting an environment	Environment	deleteEnvironment
Creating a variable	EnvVariable	CreateEnvironmentVaria-ble
Deleting a variable	EnvVariable	DeleteEnvironmentVaria-ble
Modifying a variable	EnvVariable	UpdateEnvironmentVari-able
Creating a request throttling policy	Throttle	CreateRequestThrottling-Policy
Modifying a request throttling policy	Throttle	UpdateRequestThrot-tlingPolicy
Deleting a request throttling policy	Throttle	DeleteRequestThrottling-Policy
Creating an API	Api	CreateApi
Modifying an API	Api	UpdateApi
Deleting an API	Api	DeleteApi
Publishing an API or taking an API offline	Api	CreateOrDeletePubli-shRecordForApi
Verifying the API definition	Api	CheckApis
Debugging an API	Api	DebugApi
Publishing multiple APIs or taking APIs offline	Api	BatchPublishOrOfflineA-pi
Switching API versions	Api	ChangeApiVersion
Taking an API version offline	Api	DeleteApiByVersionId
Creating a signature key	Signature	CreateSignatureKey
Modifying a signature key	Signature	UpdateSignatureKey
Deleting a signature key	Signature	DeleteSignatureKey
Binding a signature key	SignatureBinding	AssociateSignatureKey

Operation	Resource Type	Trace Name
Unbinding a signature key	SignatureBinding	DisassociateSignatureKey
Binding a request throttling policy	ThrottleBinding	AssociateRequestThrottlingPolicy
Unbinding a request throttling policy	ThrottleBinding	DisassociateRequestThrottlingPolicy
Unbinding multiple request throttling policies	ThrottleBinding	BatchDisassociateThrottlingPolicy
Creating a special request throttling configuration	ThrottleSpecial	CreateSpecialThrottlingConfiguration
Modifying a special request throttling configuration	ThrottleSpecial	UpdateSpecialThrottlingConfiguration
Deleting an excluded request throttling configuration	ThrottleSpecial	DeleteSpecialThrottlingConfiguration
Authorizing apps	AppAuth	CreateAuthorizingApps
Canceling authorization	AppAuth	CancelingAuthorization
Binding a domain name	ApiGroup	AssociateDomain
Adding a certificate to a domain name	ApiGroup	AssociateCertificate
Modifying a domain name	ApiGroup	UpdateDomain
Unbinding a domain name	ApiGroup	DisassociateDomain
Deleting a domain certificate	ApiGroup	DisassociateCertificate
Creating an access control policy	Acl	CreateAclStrategy
Modifying an access control policy	Acl	UpdateAclStrategy
Deleting an access control policy	Acl	DeleteAcl
Deleting multiple access control policies	Acl	BatchDeleteAclV2

Operation	Resource Type	Trace Name
Binding an access control policy to an API	AclBinding	CreateApiAclBinding
Unbinding an access control policy from an API	AclBinding	DeleteApiAclBinding
Unbinding multiple access control policies from APIs	AclBinding	BatchDeleteApiAclBind-ing
Creating a custom authorizer	Authorizer	CreateCustomAuthorizer
Modifying a custom authorizer	Authorizer	UpdateCustomAuthorizer
Deleting a custom authorizer	Authorizer	DeleteCustomAuthorizer
Exporting APIs	Swagger	ExportApiDefinitions
Importing APIs	Swagger	ImportApiDefinitions
Creating a VPC channel	Vpc	CreateVpcChannel
Updating a VPC channel	Vpc	UpdateVpcChannel
Deleting a VPC channel	Vpc	DeleteVpcChannel
Adding or updating backend instances	Vpc	AddingBackendInstances
Updating backend instances	Vpc	UpdateBackendInstances
Removing a backend server	Vpc	DeleteBackendInstance
Enabling backend servers	Vpc	BatchEnableMembers
Disabling backend servers	Vpc	BatchDisableMembers
Modifying VPC channel health check	Vpc	UpdateHealthCheck
Adding or updating a backend server group of a VPC channel	Vpc	CreateMemberGroup
Deleting a backend server group of a VPC channel	Vpc	DeleteMemberGroup

Operation	Resource Type	Trace Name
Updating a Backend Server Group of a VPC Channel	Vpc	UpdateMemberGroup
Creating a response for an API group	ApiGroup	CreateGatewayResponse
Modifying a response of an API group	ApiGroup	UpdateGatewayResponse
Deleting a response of an API group	ApiGroup	DeleteGatewayResponse
Modifying the response of an error type defined for an API group	ApiGroup	UpdateGatewayRespon-seType
Deleting the response of an error type defined for an API group	ApiGroup	DeleteGatewayResponse-Type
Configuring a feature for a gateway	Feature	CreateFeatureV2
Creating a dedicated gateway (Pay-per-use)	Instance	CreateInstance
Updating a dedicated gateway	Instance	UpdateInstance
Binding an EIP to a gateway or updating the EIP of a gateway	Instance	AddEip
Unbinding the EIP of a gateway	Instance	RemoveEip
Enabling public outbound access for a gateway	Instance	AddEngressEip
Updating the public outbound access bandwidth of a gateway	Instance	UpdateEngressEip
Disabling public outbound access for a gateway	Instance	RemoveEngressEip
Enabling public inbound access	Instance	AddIngressEip
Updating the public inbound access bandwidth of a gateway	Instance	UpdateIngressEip

Operation	Resource Type	Trace Name
Disabling public inbound access for a gateway	Instance	RemoveIngressEip
Deleting a dedicated gateway	Instance	DeleteInstances
Modifying the specifications of a pay-per-use gateway	Instance	CreatePostPayResizeOr-der
Accepting or rejecting a VPC endpoint connection	vpc-endpoint	AcceptOrRejectEndpoint-Connections
Adding whitelist records for a VPC endpoint service	vpc-endpoint	AddEndpointPermissions
Deleting whitelist records of a VPC endpoint service	vpc-endpoint	DeleteEndpointPermis-sions
Batch adding or deleting gateway tags	Instance	BatchCreateOrDeleteIn-stanceTags
Importing a microservice	Microservice	ImportMicroservice
Adding an SSL certificate	SslCertificate	CreateCertificate
Binding a domain name with SSL certificates	ApiGroup	BatchAssociateCerts
Unbinding the SSL certificates of a domain name	ApiGroup	BatchDisassociateCerts
Deleting an SSL certificate	SslCertificate	DeleteCertificate
Modifying an SSL certificate	SslCertificate	UpdateCertificate
Binding an SSL certificate to a domain name	Certificate	BatchAssociateDomains
Unbinding an SSL certificate from a domain name	Certificate	BatchDisassociateDo-mains
Creating a plug-in	Plugin	CreatePlugin
Modifying a plug-in	Plugin	UpdatePlugin
Deleting a plug-in	Plugin	DeletePlugin
Binding a plug-in to an API	Plugin	AttachApiToPlugin

Operation	Resource Type	Trace Name
Binding a plug-in to an API	Plugin	AttachPluginToApi
Unbinding an API from a plug-in	Plugin	DetachApiFromPlugin
Unbinding a plug-in from an API	Plugin	DetachPluginFromApi
Creating an app	App	CreateAnApp
Modifying an app	App	UpdateApp
Deleting an app	App	DeleteAppV2
Resetting an Appsecret	App	ResettingAppSecret
Verifying an app	App	CheckApp
Creating an AppCode	AppCode	CreateAppCode
Generating an AppCode	AppCode	CreateAppCodeAuto
Deleting an AppCode	AppCode	DeleteAppCode
Configuring access control settings for an app	AppAcl	UpdateAppAcl
Deleting access control settings of an app	AppAcl	DeleteAppAcl
Creating a credential quota	AppQuota	CreateAppQuota
Modifying a credential quota	AppQuota	UpdateAppQuota
Deleting a credential quota	AppQuota	DeleteAppQuota
Binding a credential quota with credentials	AppQuotaBinding	AssociateAppsForApp-Quota
Unbinding a credential quota from a credential	AppQuotaBinding	DisassociateAppQuota-WithApp

Disabling CTS

Disable CTS by following the procedure in section "Deleting a Tracker" in the *Cloud Trace Service User Guide*.

13.2 Querying Real-Time Traces

Scenarios

After you enable CTS and the management tracker is created, CTS starts recording operations on cloud resources. After a data tracker is created, the system starts recording operations on data in OBS buckets. CTS stores operation records generated in the last seven days.

This section describes how to query and export operation records of the last seven days on the CTS console.

- **Viewing Real-Time Traces in the Trace List of the New Edition**
- **Viewing Real-Time Traces in the Trace List of the Old Edition**

Viewing Real-Time Traces in the Trace List of the New Edition

1. Log in to the management console.
2. Click ☰ in the upper left corner and choose Management & Deployment > **Cloud Trace Service**. The CTS console is displayed.
3. Choose **Trace List** in the navigation pane on the left.
4. On the **Trace List** page, use advanced search to query traces. You can combine one or more filters.
   - **Trace Name**: Enter a trace name.
   - **Trace ID**: Enter a trace ID.
   - **Resource Name**: Enter a resource name. If the cloud resource involved in the trace does not have a resource name or the corresponding API operation does not involve the resource name parameter, leave this field empty.
   - **Resource ID**: Enter a resource ID. Leave this field empty if the resource has no resource ID or if resource creation failed.
   - **Trace Source**: Select a cloud service name from the drop-down list.
   - **Resource Type**: Select a resource type from the drop-down list.
   - **Operator**: Select one or more operators from the drop-down list.
   - **Trace Status**: Select **normal**, **warning**, or **incident**.
     - **normal**: The operation succeeded.
     - **warning**: The operation failed.
     - **incident**: The operation caused a fault that is more serious than the operation failure, for example, causing other faults.
   - Time range: Select **Last 1 hour**, **Last 1 day**, or **Last 1 week**, or specify a custom time range.
5. On the **Trace List** page, you can also export and refresh the trace list, and customize the list display settings.

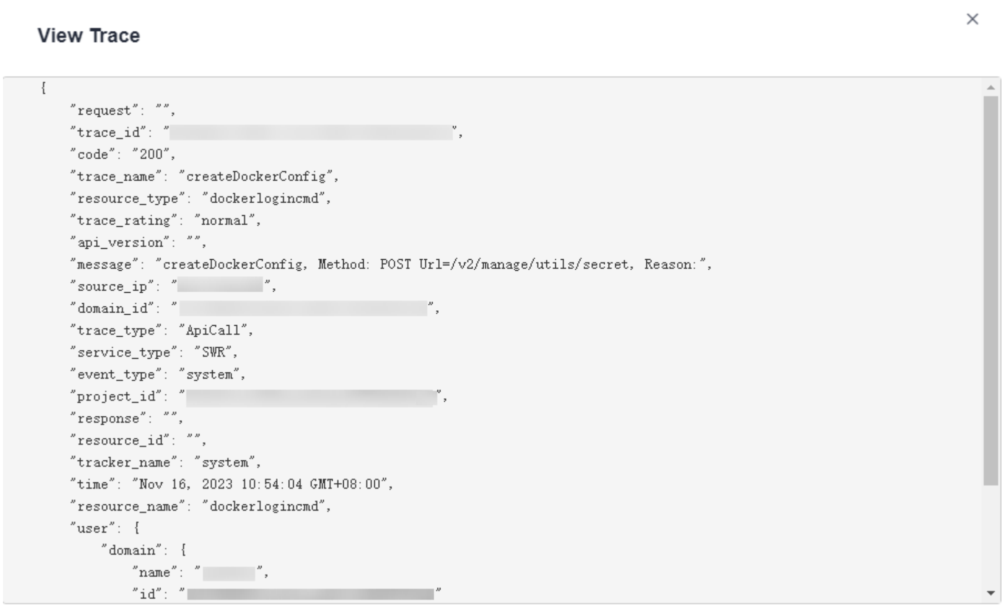
- Enter any keyword in the search box and press Enter to filter desired traces.
- Click **Export** to export all traces in the query result as an .xlsx file. The file can contain up to 5000 records.
- Click to view the latest information about traces.
- Click to customize the information to be displayed in the trace list. If **Auto wrapping** is enabled ( ), excess text will move down to the next line; otherwise, the text will be truncated. By default, this function is disabled.

6. For details about key fields in the trace structure, see section "Trace References" > "Trace Structure" and section "Trace References" > "Example Traces".
7. (Optional) On the **Trace List** page of the new edition, click **Go to Old Edition** in the upper right corner to switch to the **Trace List** page of the old edition.

Viewing Real-Time Traces in the Trace List of the Old Edition

1. Log in to the management console.
2. Click in the upper left corner and choose **Management & Deployment** > **Cloud Trace Service**. The CTS console is displayed.
3. Choose **Trace List** in the navigation pane on the left.
4. Each time you log in to the CTS console, the new edition is displayed by default. Click **Go to Old Edition** in the upper right corner to switch to the trace list of the old edition.
5. Set filters to search for your desired traces. The following filters are available:
   - **Trace Type**, **Trace Source**, **Resource Type**, and **Search By**: Select a filter from the drop-down list.
     - If you select **Resource ID** for **Search By**, specify a resource ID.
     - If you select **Trace name** for **Search By**, specify a trace name.
     - If you select **Resource name** for **Search By**, specify a resource name.
   - **Operator**: Select a user.
   - **Trace Status**: Select **All trace statuses**, **Normal**, **Warning**, or **Incident**.
   - Time range: You can query traces generated during any time range in the last seven days.
   - Click **Export** to export all traces in the query result as a CSV file. The file can contain up to 5000 records.
6. Click **Query**.
7. On the **Trace List** page, you can also export and refresh the trace list.
   - Click **Export** to export all traces in the query result as a CSV file. The file can contain up to 5000 records.
   - Click to view the latest information about traces.
8. Click on the left of a trace to expand its details.

9. Click **View Trace** in the **Operation** column. The trace details are displayed.


View Trace

```
{
    "request": "",
    "trace_id": "[redacted]",
    "code": "200",
    "trace_name": "createDockerConfig",
    "resource_type": "dockerlogincmd",
    "trace_rating": "normal",
    "api_version": "",
    "message": "createDockerConfig, Method: POST Url=/v2/manage/utils/secret, Reason:",
    "source_ip": "[redacted]",
    "domain_id": "[redacted]",
    "trace_type": "ApiCall",
    "service_type": "SWR",
    "event_type": "system",
    "project_id": "[redacted]",
    "response": "",
    "resource_id": "",
    "tracker_name": "system",
    "time": "Nov 16, 2023 10:54:04 GMT+08:00",
    "resource_name": "dockerlogincmd",
    "user": {
        "domain": {
            "name": "[redacted]",
            "id": "[redacted]"
```

10. For details about key fields in the trace structure, see section "Trace References" > "Trace Structure" and section "Trace References" > "Example Traces" in the *CTS User Guide*.
11. (Optional) On the **Trace List** page of the old edition, click **New Edition** in the upper right corner to switch to the **Trace List** page of the new edition.

14 FAQs

14.1 Common FAQs

API Creation

- How Do I Set the Backend Address If I Will Not Use a VPC Channel (or Load Balance Channel)?
- How Can I Configure the Backend Service Address?
- Can I Specify a Private Network Load Balancer Address for the Backend Service?
- Can I Specify the Backend Address as a Subnet IP Address?
- Can I Bind Private Domain Names for API Access?

API Calling

- What Are the Possible Causes for an API Calling Failure?
- What Should I Do If an Error Code Is Returned During API Calling?
- What Should I Do If "The API does not exist or has not been published in the environment." Is Displayed?
- Why Am I Seeing the Message "No backend available"?
- What Are the Possible Causes If the Message "Backend unavailable" or "Backend timeout" Is Displayed?

API Authentication

- Does APIG Support HTTPS Two-Way Authentication?
- How Do I Call an API That Does Not Require Authentication?

API Control Policies

- Can I Configure the Maximum Number of Concurrent Requests?
- Does APIG Have Bandwidth Limits?
- How Do I Provide an Open API to Specific Users?

- **How Do I Exclude a Specific IP Address for Identity Authentication of an API?**

API Import and Export

- **Why Does API Import Fail?**
- **Does APIG Provide a Template for Importing APIs from Swagger Files?**

14.2 API Creation

14.2.1 Why Can't I Create APIs?

The creation of APIs is free of charge. If you cannot create APIs, your account must be in arrears.

For details, see section "Billing" in the *API Gateway User Guide*.

14.2.2 How Do I Define Response Codes for an API?

There are two types of responses:

- Gateway response codes: returned by the gateway for API requests that are throttled, denied, or failed in authentication. For details about these response codes, see section "Creating a Gateway Response" in the *API Gateway User Guide*.
- Backend service responses: defined by backend API services (API providers) and transparently transmitted by APIG.

14.2.3 How Do I Specify the Host Port for a VPC Channel (or Load Balance Channel)?

Use the port of the API backend service.

For details about how to configure the API backend, see section "Creating an API" in the *API Gateway User Guide*.

14.2.4 How Do I Set the Backend Address If I Will Not Use a VPC Channel (or Load Balance Channel)?

Specify the backend address as a public domain name or a public IP address, such as the Elastic IP (EIP) of an Elastic Cloud Server (ECS). To do this, enable public outbound access for the gateway.

Use a private network IP address, not a private network domain name.

14.2.5 How Can I Configure the Backend Service Address?

Configure the backend service address as an ECS EIP, or the public IP address or domain name of your own server.

For details about how to configure the API backend, see section "Creating an API" in the *API Gateway User Guide*.

14.2.6 Can I Specify a Private Network Load Balancer Address for the Backend Service?

- For dedicated gateways, you can use private network load balancer addresses.
- Alternatively, you can use the EIP bound to a public network load balancer.

14.2.7 Can I Specify the Backend Address as a Subnet IP Address?

If you use a dedicated gateway, you can specify either an IP address that belongs to the same subnet where the gateway is deployed, or the private address of a local data center connected to the gateway through Direct Connect.

Unsupported network segments:

- 0.0.0.0/8
- 10.0.0.0/8
- 100.125.0.0/16
- 127.0.0.0/8
- 169.254.0.0/16
- 172.16.0.0/12
- 192.0.0.0/24
- 192.0.2.0/24
- 192.88.99.0/24
- 192.168.0.0/16
- 198.18.0.0/15
- 198.51.100.0/24
- 203.0.113.0/24
- 224.0.0.0/4
- 240.0.0.0/4
- 255.255.255.255/32

14.2.8 Does APIG Support Multiple Backend Endpoints?

Yes

APIG supports the configuration of multiple backend endpoints through a VPC channel (also called "load balance channel"). You can add multiple cloud servers to each VPC channel.

For details, see section "Creating a VPC Channel (Load Balance Channel)" in the *API Gateway User Guide*.

14.2.9 What Should I Do After Applying for an Independent Domain Name?

If you are using a dedicated gateway, add an A record that points the independent domain name to the inbound access address of the gateway. You can bind five independent domain names to an API group but can bind each independent domain name only to one API group.

NOTE

To use a public domain name, add an A record (dedicated gateway) in Domain Name Service (DNS).

To use a private domain name, add an A record (dedicated gateway) in the DNS service and associate the domain name with the VPC in which your backend service is located.

14.2.10 Can I Bind Private Domain Names for API Access?

In a dedicated gateway, you can add a private domain name (filing not required), and add an A record to point the domain name to the inbound access address of the gateway.

14.2.11 Why Does an API Failed to Be Called Across Domains?

1. Ensure that CORS has been enabled for the API.

   Go to the API details page, click **Edit**, and check whether CORS is enabled. If it is not, enable it.
2. Check whether an API with the OPTIONS method has been created. Only one such API is required for each API group.

   NOTE

   Parameters are as follows:

   API Group: The same group to which the API with CORS enabled belongs.

   Method: Select **OPTIONS**.

   Protocol: The same protocol used by the API with CORS enabled.

   Path: Same as or prefixally matching the request path set for the API with CORS enabled.

   Matching: Select **Prefix match**.

   Authentication Mode: **None** means all users will be granted access. It is not recommended.

   CORS: Enable this option.

14.3 API Calling

14.3.1 What Are the Possible Causes for an API Calling Failure?

Network

API calling failures may occur in three scenarios: within a VPC, between VPCs, and on a public network.

- Within a VPC: Check whether the domain name is the same as that automatically allocated for the API.
- Between VPCs: Check whether the two VPCs are connected. If they are not connected, create a VPC peering connection to connect the two VPCs.

For details about how to create and use VPC peering connections, see section "VPC Peering Connection" in the *Virtual Private Cloud User Guide* or section "Exposing Backend Services Across VPCs" in the *API Gateway Best Practices*.

- On a public network:
  - The API is not bound with an EIP and does not have a valid address for public network access.

    Bind an EIP to the API and try again. For details, see section "Buying a Gateway" in the *API Gateway User Guide*.
  - The inbound rules are incorrectly configured.

    For details about how to configure inbound rules, see section "Buying a Gateway" in the *API Gateway User Guide*.
  - The request header "host:*Group domain name*" is not added when you call the API. Add the request header and try again.

Domain Name

- Check whether the domain name bound to the API group to which the API belongs has been successfully licensed and can be resolved.
- Check whether the domain name has been bound to the correct API group.
- The subdomain name (debugging domain name) automatically allocated to the API group is accessed too many times. The subdomain name can be accessed only 1000 times a day. It is unique and cannot be modified. Add independent domain names for the group to make the APIs in the group accessible.

API Publishing

Check whether the API has been published. If the API has been modified, publish it again. If the API has been published to a non-RELEASE environment, specify the **X-Stage** header as the environment name.

API Authentication

If the API uses app authentication, check whether the AppKey and AppSecret used to call the API are correct.

API Control Policies

- Check whether the access control policy bound to the API is correct.
- Check whether the request throttling limit of the API has been reached. If no request throttling policy is created for an API, the API can be accessed 200 times per second by default. To change this limit of dedicated gateways, go to the **Gateway Information** page, click the **Parameters** tab, and modify the **ratelimit_api_limits** parameter.

14.3.2 What Should I Do If an Error Code Is Returned During API Calling?

If an error code is returned when you call your own APIs, see**Error Codes**.

If an error code is returned when you manage your APIs,see section "Error Codes" in the *API Gateway API Reference*.

14.3.3 Why Am I Seeing the Error Message "414 Request URI too large" When I Call an API?

The request URL (including request parameters) is too long. Place the request parameters in the request body and try again.

For details about API calling errors, see section "Error Codes" in the *API Gateway User Guide*.

14.3.4 What Should I Do If "The API does not exist or has not been published in the environment." Is Displayed?

If an open API in APIG failed to be called, troubleshoot the failure by performing the following operations:

1. The domain name, request method, or path used for calling the API is incorrect.
   - For example, an API created using the POST method is called with GET.
   - Missing a slash (/) in the access URL will lead to a failure in matching the URL in the API details. For example, URLs **http://7383ea59c0cd49a2b61d0fd1d351a619.apigw.region.cloud.com/test/** and **http://7383ea59c0cd49a2b61d0fd1d351a619.apigw.region.cloud.com/test** represent two different APIs.
2. The API has not been published. APIs can be called only after they have been published in an environment. For details, see section "Publishing an API" in the *API Gateway User Guide*. If the API has been published in a non-production environment, check whether the **X-Stage** header in the request is the name of the environment.
3. The domain name is resolved incorrectly. If the domain name, request method, and path for calling the API are correct and the API has been published in an environment, the API may not be correctly resolved to the group to which the API belongs. For example, if you have multiple API groups and each group has an independent domain name, the API may be called using the independent domain name of another group. Ensure that the API is being called using the correct domain name.
4. Check whether the API allows OPTIONS cross-region requests. If yes, enable cross-origin resource sharing (CORS) for the API, and create an API that uses the OPTIONS method. For details, see section "CORS" in the *API Gateway User Guide*.

14.3.5 Why Am I Seeing the Message "No backend available"?

- Check whether the backend service is accessible, and modify the backend service if it is inaccessible.
- Check the ECS security group configurations of the backend service and verify that the required port has been enabled.

- Check whether the backend service address is a public IP address. If yes, enable outbound access on the **Gateways** > **Access Console** > **Gateway Information** page.
- Check whether ACL configurations of the VPC restrict the communication between the API gateway and the subnet where the backend service is located.
- If you use a VPC channel, check whether the service port, health check port, and backend servers of the VPC channel have been correctly configured.

  .

14.3.6 What Are the Possible Causes If the Message "Backend unavailable" or "Backend timeout" Is Displayed?

The following table lists the possible causes if a backend service fails to be invoked or the invocation times out.

Possible Cause	Solution
The backend service address is incorrect.	Change the backend service address in the API definition. If the domain name is used, ensure that the domain name can be correctly resolved to the IP address of the backend service.
The timeout duration is incorrect. If a backend service fails to return a response within the configured timeout duration, APIG displays a message indicating that the backend service fails to be invoked.	Increase the backend timeout duration in the API definition.
If the backend address is an ECS address, the security group to which the ECS belongs may block the request in the inbound or outbound direction.	Check the security group to which the ECS belongs and ensure that the inbound and outbound port rules and protocols of this security group are correct.
The request protocol is incorrect. For example, the backend service uses HTTP, but HTTPS is selected on APIG.	Ensure that the protocol of the created API is the same as that of the backend service.
The backend service URL is unreachable.	Check the URL.

14.3.7 Why Am I Seeing the Message "Backend domain name resolution failed" When a Backend Service Is Called?

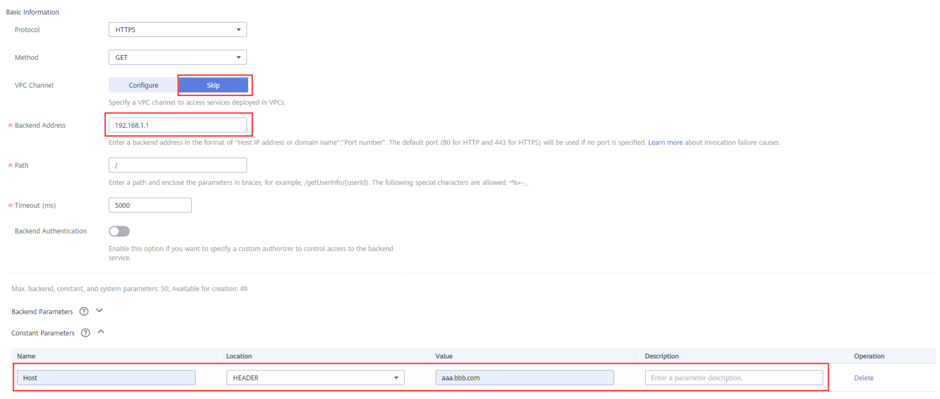
An error message indicating a domain name resolution failure is displayed when the backend service is called, although private domain name resolution is completed for the VPC where the API gateway is located.

Possible Cause

The VPC of the API gateway is isolated from that of the backend service. Private domain names can be resolved only for the VPC of the backend service.

Solution

- Method 1: When creating an API, set **Backend Address** to a public network domain name.
- Method 2: When creating an API, do not use a VPC channel (load balance channel). Instead, set **Backend Address** to the backend service IP address, and add a constant parameter to specify the **Host** field in the header.



Basic Information

Protocol	HTTPS
Method	GET
VPC Channel	Configure [✓ Skip]
	Specify a VPC channel to access services deployed in VPCs.
* Backend Address	192.168.1.1
	Enter a backend address in the format of "Host IP address or domain name":"Port number". The default port (80 for HTTP and 443 for HTTPS) will be used if no port is specified. Learn more about invocation failure causes.
* Path	/
	Enter a path and enclose the parameters in braces, for example, /getUserInfo/{userId}. The following special characters are allowed: *%+-_.
* Timeout (ms)	5000
Backend Authentication	
	Enable this option if you want to specify a custom authorizer to control access to the backend service.

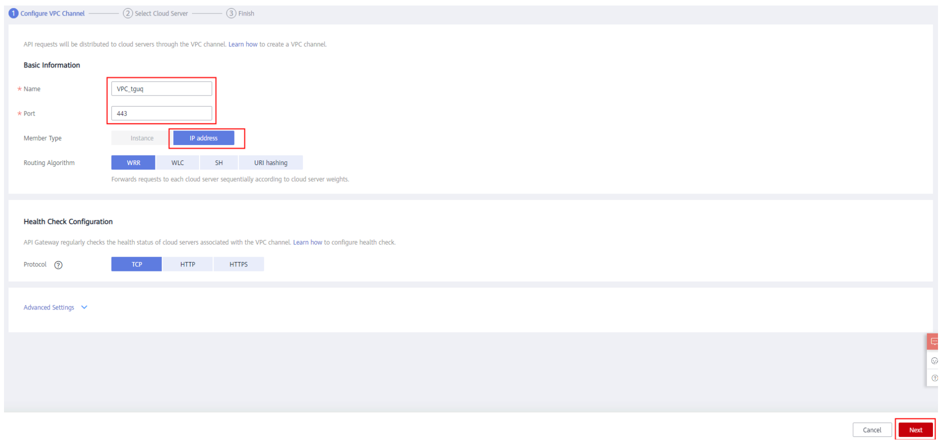
Max. backend, constant, and system parameters: 50; Available for creation: 49

Backend Parameters

Constant Parameters

Name	Location	Value	Description	Operation
Host	HEADER	aaa.bbb.com	Enter a parameter description.	Delete

- Method 3: When creating an API, specify a VPC channel (load balance channel).
  a. Create a VPC channel (load balance channel).



API requests will be distributed to cloud servers through the VPC channel. Learn how to create a VPC channel.

Basic Information

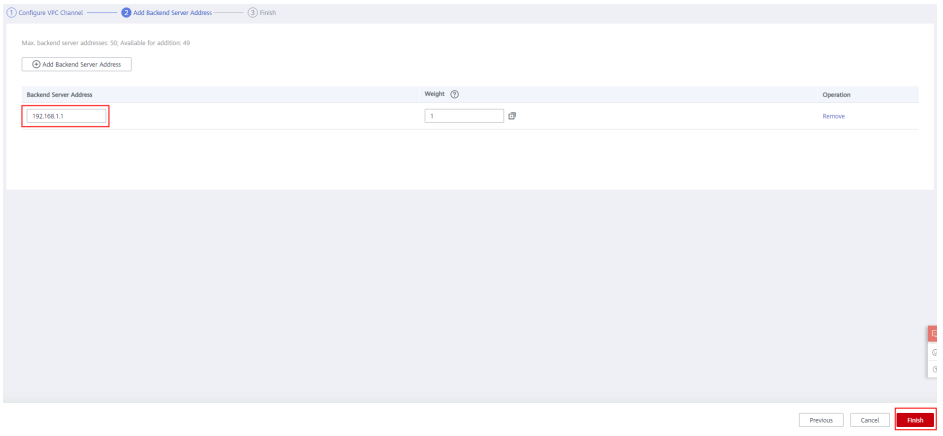
* Name	VPC_tguq
* Port	443
Member Type	Instance [✓ IP address]
Routing Algorithm	[✓ WRR] WLC SH URI hashing
	Forwards requests to each cloud server sequentially according to cloud server weights.

Health Check Configuration

API Gateway regularly checks the health status of cloud servers associated with the VPC channel. Learn how to configure health check.

Protocol	[✓ TCP] HTTP HTTPS

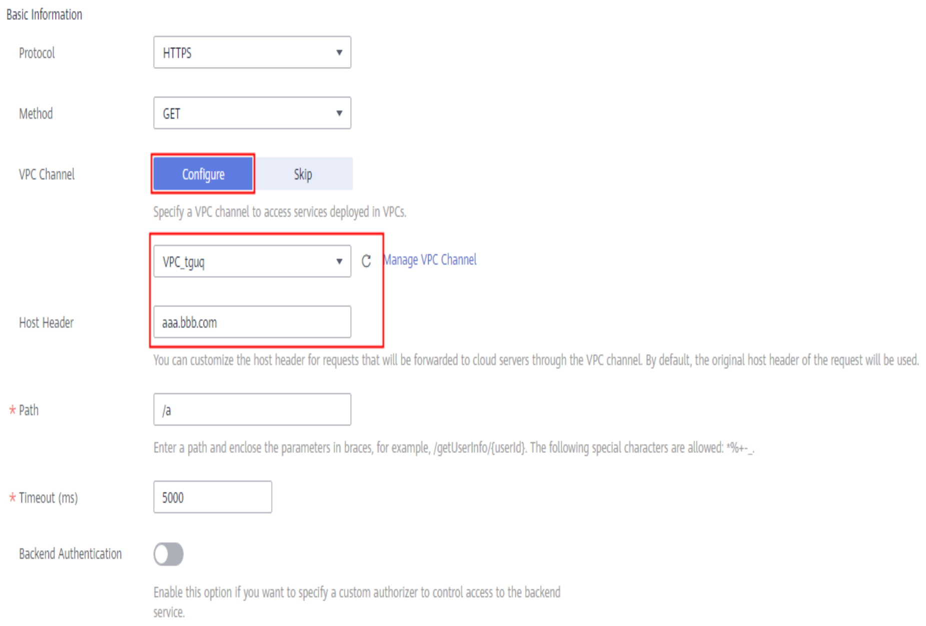
  b. Add the backend service address.



Max. backend server addresses: 50; Available for addition: 49

Backend Server Address	Weight	Operation
192.168.1.1	1	Remove

c. When creating an API, select the VPC channel (load balance channel) and configure a custom header.



Protocol: HTTPS

Method: GET

VPC Channel: Configure

Specify a VPC channel to access services deployed in VPCs.

VPC_tguq

Host Header: aaa.bbb.com

You can customize the host header for requests that will be forwarded to cloud servers through the VPC channel. By default, the original host header of the request will be used.

* Path: /a

Enter a path and enclose the parameters in braces, for example, /getUserInfo/{userId}. The following special characters are allowed: *%+-_.

* Timeout (ms): 5000

Backend Authentication

Enable this option if you want to specify a custom authorizer to control access to the backend service.

14.3.8 Why Doesn't Modification of the backend_timeout Parameter Take Effect?

Problem Description

Modification of the **backend_timeout** parameter in gateways does not take effect.

Possible Causes

The **Timeout (ms)** parameter on the **Define Backend Request** page is not modified.

Solution

Log in to the APIG console, go to the API details page, click **Edit**, and modify the **Timeout (ms)** parameter on the **Define Backend Request** page.

14.3.9 How Do I Switch the Environment for API Calling?

By default, the API in the RELEASE environment is called.

If you want to call the same API in another environment, add the request header **X-Stage** to specify the environment name.

14.3.10 What Is the Maximum Size of an API Request Package?

Dedicated gateway: APIG forwards only API requests whose body is no larger than 12 MB. If your gateway will receive requests with a body larger than 12 MB, modify the **request_body_size** parameter on the gateway details page. This parameter indicates the maximum request body size allowed. The value ranges from 1 MB to 9536 MB.

14.3.11 How Do I Perform App Authentication in iOS System?

APIG provides SDKs and demos in multiple languages, such as Java, Python, C, PHP, and Go, for app authentication.

To use Objective-C (for iOS) or other languages, see **Developer Guide > Calling APIs Through App Authentication > App Authentication Principle**.

14.3.12 Why Can't I Create a Header Parameter Named x-auth-token for an API Called Through IAM Authentication?

The header parameter **x-auth-token** has already been defined in APIG.

To use this parameter to call an API, add the parameter and its value to the request header.

14.3.13 App (Credential) FAQs

How many apps (credentials) can I create?

You can create a maximum of 50 apps (credentials).

How do I isolate the calling information among the third parties that call the same API through app authentication?

Create multiple apps (credentials) for different third parties and bind the apps (credentials) to the same API.

Are there any restrictions on the maximum number of third parties that can call the same app through app authentication?

No restrictions.

Do I need to create an app (credential) for an API so that it can be called through app authentication?

Yes, you need to create an app (credential) and bind it to the API. After the app (credential) is created, an AppKey and AppSecret are automatically created. Provide the AppKey and AppSecret for third parties to call the API.

How can an API be called by third parties through app authentication?

Provide third parties with the AppKey and AppSecret of the app you have created for accessing the API. The third parties then can use the AppKey and AppSecret to call the API through an SDK. For details about how to use an SDK, see **Developer Guide > Calling APIs Through App Authentication**.

14.3.14 Can Mobile Apps Call APIs?

Yes, mobile apps can call APIs.

In app authentication mode, the AppKey and AppSecret of a mobile app are replaced with those in the relevant SDK to sign the app.

14.3.15 Can Applications Deployed in a VPC Call APIs?

Yes, applications deployed in a VPC can call APIs by default. If domain name resolution fails, configure a DNS server on the current endpoint by following the instructions in **Configuring an Intranet DNS Server**. After the configuration, applications deployed in the VPC can call APIs.

Configuring an Intranet DNS Server

To configure a DNS server, specify its IP address in the **/etc/resolv.conf** file.

The IP address of the intranet DNS server depends on which region you are located in. Find the IP address of the intranet DNS server in your region from the private DNS server addresses mentioned in "FAQs" of the *Domain Name Service User Guide*.

Add an intranet DNS server with either of the following two methods:

- Method 1: Modify the subnet information of the VPC.
- Method 2: Edit the **/etc/resolv.conf** file.

NOTE

The intranet DNS server configurations become invalid after the ECS restarts, and the intranet DNS server must be configured again. Therefore, method 1 is recommended.

Method 1

Perform the following procedure to add a DNS server IP address to the subnet configurations of the ECS in the VPC.

Step 1 Log in to the management console.

Step 2 Click in the upper left corner to select a region.

Step 3 In the service list, choose **Compute** > **Elastic Cloud Server**.

Step 4 Click the name of the ECS you want to use.

Step 5 On the **Network Interfaces** tab page, click to view the subnet name.

Step 6 On the **Summary** tab page, view the VPC name.

Step 7 Click the VPC name to visit the VPC console.

Step 8 Choose **Subnets** in the left navigation pane.

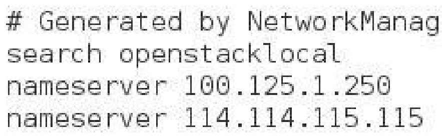
Step 9 Locate the subnet mentioned in **Step 5** and click the subnet name.

Step 10 Change the DNS server address of the subnet and click **OK**.

For example, change the address to **100.125.1.250**.

Step 11 Restart the ECS. Check that the **/etc/resolv.conf** file contains the IP address of the DNS server to be configured, and the IP address is less than those of all other DNS servers.

The following figure shows the IP address **100.125.1.250** of the DNS server to be configured.



```
# Generated by NetworkManager
search openstacklocal
nameserver 100.125.1.250
nameserver 114.114.115.115
```

NOTE

Modifying the subnet information of a VPC will affect all ECSs created using the subnet.

----End

Method 2

Add the IP address of the intranet DNS server to the **/etc/resolv.conf** file.

For example, if you are located in **region01**, add an intranet DNS server of IP address **100.125.1.250** to the **/etc/resolv.conf** file.

NOTE

- The IP address of the new DNS server must be less than those of all other DNS servers.
- The DNS configurations take effect immediately after the **/etc/resolv.conf** file is saved.

14.3.16 Does APIG Support WebSocket Data Transmission?

Yes.

When creating an API, you can select HTTP, HTTPS, or HTTP&HTTPS. HTTP is equivalent to WebSocket (ws), and HTTPS is equivalent to WebSocket Secure (wss).

14.3.17 Does APIG Support Persistent Connections?

Yes.

But you should use persistent connections properly to avoid occupying too many resources.

14.3.18 How Will the Requests for an API with Multiple Backend Policies Be Matched and Executed?

If multiple backend policies are configured for an API, APIG will match the backend policies in sequence. If an API request matches one of the backend policies, APIG immediately forwards the request to the corresponding backend and stops matching.

If no backend policy is matched, the API request is forwarded to the default backend server.

14.3.19 Is There a Limit on the Size of the Response to an API Request?

No.

But there is a limit on the size of the request body. For details, see the **request_body_size** parameter in the *API Gateway User Guide*.

14.3.20 How Can I Access Backend Services over Public Networks Through APIG?

Enable public access for the relevant gateway to allow external services to call APIs.

If you encounter a network problem when calling APIs, see **What Are the Possible Causes for an API Calling Failure?**

14.4 API Authentication

14.4.1 Does APIG Support HTTPS Two-Way Authentication?

Dedicated gateway: Yes.

- Backend two-way authentication: When creating an API, enable two-way authentication for the backend service. For details, see the description about **Two-Way Authentication** in **Creating an API**.

14.4.2 How Do I Call an API That Does Not Require Authentication?

To call APIs that do not require authentication, construct standard HTTP requests and send them to APIG.

NOTE

APIG **transparently transmits** requests to call an API that does not require authentication to the backend service. If you want requests to be authenticated on the API backend service, you can set **Security Authentication** to **None**. The API caller transfers the fields required for authentication to the backend service, and the backend service performs authentication.

14.4.3 Which TLS Versions Does APIG Support?

APIG supports TLS 1.1 and TLS 1.2, but does not support TLS 1.0 or TLS 1.3.

For details, see section "Binding a Domain Name" in the *API Gateway User Guide*.

14.4.4 Does APIG Support Custom Authentication?

Yes.

For details, see section "Custom Authorizers" in the *API Gateway User Guide*.

14.4.5 Will the Request Body Be Signed for Security Authentication?

Yes. The request body is another element that needs to be signed in addition to the mandatory request header parameters. For example, when an API used to upload a file using the POST method is called, the hash value of the file to upload is calculated to generate a signature.

For details about signatures, see section "App Authentication" in the *API Gateway Developer Guide*.

14.4.6 Common Errors Related to IAM Authentication Information

You may encounter the following errors related to IAM authentication information:

- **Incorrect IAM authentication information: AK access failed to reach the limit,forbidden**
- **Incorrect IAM authentication information: decrypt token fail**
- **Incorrect IAM authentication information: Get secretKey failed**

Incorrect IAM authentication information: AK access failed to reach the limit,forbidden

```
{
  "error_msg": "Incorrect IAM authentication information: AK access failed to reach the
limit,forbidden." ......
  "error_code": "APIG.0301",
  "request_id": "******"
}
```

Possible Causes

- The AK/SK signature calculation is incorrect.
- The AK and SK do not match.
- AK/SK authentication fails for more than five consecutive times, and the AK/SK pair is locked for five minutes. (Authentication requests are rejected within this period).
- An expired token is used for token authentication.

Incorrect IAM authentication information: decrypt token fail

```
{
  "error_msg": "Incorrect IAM authentication information: decrypt token fail",
  "error_code": "APIG.0301",
  "request_id": "******"
}
```

Possible Cause

The token cannot be parsed for IAM authentication of the API.

Solution

- Check whether the obtained token is the token of the corresponding IAM account.
- Check whether the token is correct.
- Check whether the token has been obtained in the environment where the API is called.

Incorrect IAM authentication information: Get secretKey failed

```
{
"error_msg": "Incorrect IAM authentication information: Get secretKey failed,ak:******,err:ak not exist",
"error_code": "APIG.0301",
"request_id": "******"
}
```

Possible Cause

The AK used for IAM authentication of the API does not exist.

Solution

Check whether the AK is correct.

14.4.7 What Should I Do If the App Authentication Information Is Incorrect?

You may encounter the following errors related to app authentication information:

- **Incorrect app authentication information: app not found, appkey xxx**
- **Incorrect app authentication information: signature expired**

Incorrect app authentication information: app not found, appkey xxx

```
{
  "error_msg": "Incorrect app authentication information: app not found, appkey 
01177c425f71487ea362ba84dc4abe5e1",
  "error_code": "APIG.0303",
  "request_id": "a5322eb89048eb41d705491a76a05aca"
}
```

Possible Causes

The AppKey is incorrect.

Solution

Step 1 In the navigation pane of the APIG console, choose **API Management** > **Credentials**.

Step 2 Click the corresponding credential name to go to the details page.

Step 3 Check the **Key** and reconfigure the AppKey.

----End

Incorrect app authentication information: signature expired

```
{
  "error_msg": "Incorrect app authentication information: signature expired, signature
time:20230527T000431Z,server time:20230527T020608Z",
  "error_code": "APIG.0303",
  "request_id": "fd6530a01c09807640189e65e837b8ad"

}
```

Possible Causes

The difference between the client's signature timestamp **x-sdk-date** and the APIG server's time exceeds 15 minutes.

Solution

Check whether the time on the client is correct.

14.5 API Control Policies

14.5.1 Request Throttling

14.5.1.1 Can I Configure the Maximum Number of Concurrent Requests?

No,

but you can limit the maximum number of API calls allowed within a specific period of time.

14.5.1.2 Is the Restriction of 1000 Requests per Day to a Subdomain Name (Debugging Domain Name) Applied to Enterprise Accounts?

Yes.

For details about subdomain names (debugging domain names), see section "Binding a Domain Name" in the *API Gateway User Guide*.

14.5.1.3 Does APIG Have Bandwidth Limits?

Dedicated gateways have bandwidth limits. When you create a dedicated gateway, you can set the bandwidth for public inbound and outbound access.

14.5.1.4 Why Doesn't a Request Throttling Policy Take Effect?

- API call limit or source IP address request limit of the policy does not take effect.

  Check whether the policy has been bound to an API.

- User request limit of the policy does not take effect.
  Check whether the API bound with the policy uses app or IAM authentication.
- App (credential) request limit of the policy does not take effect.
  Check whether the API bound with the policy uses app authentication.

14.5.2 Access Control

14.5.2.1 How Do I Provide an Open API to Specific Users?

You can provide an open API to specific users in either of the following ways:

- Select app authentication when you create the API, and share the AppKey and AppSecret with the target users.
- Configure an access control policy to allow access from specific IP addresses or account names, and bind the access control policy to the API.

14.5.2.2 How Do I Exclude a Specific IP Address for Identity Authentication of an API?

You can choose either of the following solutions:

- Solution 1: Create an API that does not require authentication, and configure an access control policy to whitelist the IP address.
- Solution 2: Create two APIs, one that uses IAM or app authentication and one that does not require authentication, and configure an access control policy to whitelist the IP address for the API that does not require authentication.

14.5.2.3 Are Client IP Addresses Verified for Access Control?

Not necessarily.

In APIG, access control is based on the value of **$remote_addr**. **$remote_addr** indicates a client IP address and is determined by the access mode. If a client accesses APIG without using any proxy, **remote_addr** is the client's IP address. If a client accesses APIG using a proxy, the client first accesses the proxy, and the proxy then forwards the request to APIG. In this case, **remote_addr** is the proxy's IP address.

14.6 API Publishing

14.6.1 Do I Need to Publish an API Again After Modification?

Yes.

After you modify the parameters of a published API, you must publish the API again to synchronize the modifications to the environment.

For details, see section "Publishing an API" in the *API Gateway User Guide*.

14.6.2 Can I Access an API Published in a Non-RELEASE Environment?

Yes. To access an API published in a non-RELEASE environment, add the **x-stage** header to the API request.

Example:

```
r.Header.Add("x-stage", "RELEASE")
```

You can also refer to examples of section "Quickly Opening and Calling APIs" in the "Getting Started" chapter of the *API Gateway User Guide*.

14.6.3 Can I Invoke Different Backend Services by Publishing an API in Different Environments?

Yes, you can invoke different backend services by publishing an API in different environments while specifying environment variables and backend parameters.

For details about environment variables, see section "Creating an Environment Variable" in the *API Gateway User Guide*.

14.6.4 Can I Specify an Environment for API Debugging?

No.

APIG debugs APIs in a specific debugging environment. After debugging is completed, you need to publish your API in an environment, and use code or Postman to add the **X-Stage** header to specify the environment where you want to call the API.

14.7 API Import and Export

14.7.1 Why Does API Import Fail?

Possible cause 1: The number of APIs exceeds the maximum allowed limit for a single import. For more APIs (300), import them in batches or submit a service ticket to increase the limit.

Possible cause 2: Parameters are incorrect. Check and rectify the parameters. You are advised to create an API on the APIG console, export it, and then use it as a template for importing APIs.

Possible cause 3: The YAML file is in incorrect format. Check and modify the file.

Possible cause 4: The local proxy network has restrictions. Change the network environment.

Possible cause 5: The header of the API request contains **X-Auth-Token**. Remove **X-Auth-Token** from the header.

14.7.2 Does APIG Provide a Template for Importing APIs from Swagger Files?

The template is being developed.

Currently, you can configure one or two APIs in APIG, and then export them to use as templates.

14.8 API Security

14.8.1 How Can I Protect My APIs?

- Identity authentication

  Configure IAM or App authentication for APIs to prevent malicious calling.
- Access control policies

  Configure a whitelist or blacklist of IP addresses/IP address ranges or accounts for APIs to secure access.
- Request throttling policies

  By default, an API can be called up to 200 times per second. If your backend service does not support this access rate, decrease the quota accordingly.

14.8.2 How Do I Ensure the Security of Backend Services Invoked by APIG?

You can ensure the security of backend services invoked by APIG by using the following methods:

- Bind signature keys to APIs

  After a signature key is bound to an API, APIG adds signature information to each request sent to the backend service. The backend service calculates the signature information in each request and checks whether the signature information is consistent with that on APIG.
- Encrypt requests using HTTPS

  Ensure that the required SSL certificate exists.
- Perform backend authentication

  Enable security authentication for backend services of the desired APIs to process only API requests that carry correct authentication information.

14.8.3 Can I Control Access to the Private IP Addresses of the ECSs in a VPC Channel (or Load Balance Channel)?

No.

14.9 Other FAQs

14.9.1 What Are the Relationships Between an API, Environment, and App (Credential)?

An API can be published in different environments, such as RELEASE (online environment) and BETA (test environment).

An app (credential) refers to the identity of an API caller. After you create an app (credential), the system automatically generates a key and secret for authenticating the app (credential). After an API is published and assigned to an app (credential), the owner of the app (credential) can call the API.

After publishing an API in different environments, you can define different request throttling policies and authorize different apps (credentials) to call the API. For example, during the test process, API v2 is published in the BETA environment and authorized to test apps (credentials). API v1 is stable and can be authorized to all users or apps (credentials) in the RELEASE environment.

14.9.2 How Can I Use APIG?

You can use APIG to manage and call APIs in the following ways:

- Management console, a web-based service management platform

  If you have already registered an account, log in to the management console, click ☰ in the upper left corner, and choose **APIG**.

  For details about the functions and operations of the APIG console, see the *API Gateway User Guide*.
- Java, Go, Python, JavaScript, C#, PHP, C++, C, and Android SDKs

  Download an SDK and use it to call APIs. For details, see the *API Gateway Developer Guide*.

14.9.3 What SDK Languages Does APIG Support?

APIG supports Java, Go, Python, C#, JavaScript, PHP, C++, C, and Android SDKs.

For details about SDKs, see the *API Gateway Developer Guide*.

14.9.4 Can I Upload Files Using the POST Method?

Yes.

If you are using dedicated gateways, configure the maximum request body size allowed by setting the **request_body_size** parameter. The value ranges from 1 MB to 9536 MB.

NOTE

Currently, only the request body can be transparently transmitted.

14.9.5 What Are the Error Messages Returned by APIG Like?

When receiving an API request, APIG returns a response. A similar response body is as follows:

```
{
    "error_code": "APIG.0101",
    "error_msg": "API does not exist or is not published in the environment.",
    "request_id": "acbc548ac6f2a0dbdb9e3518a7c0ff84"
}
```

- **"error_code"**: error code
- **"error_msg"**: description of the error

15 Change History

Table 15-1 Change history

Released On	Description
2024-10-09	This issue incorporates the following changes: ● Updated **Comparing Versions**. ● Added **Importing a CCE Workload**. ● Added **Creating a gRPC API**. ● Added **Cloning an API**. ● Added **Importing and Exorting APIs**. ● Added **Third-Party Authorizer**. ● Added **Managing Tags**. ● Added **Managing VPC Endpoints**. ● Added **Modifying Specifications**. ● Added **What Should I Do If the App Authentication Information Is Incorrect?**. ● Added **Are Client IP Addresses Verified for Access Control?**.
2023-05-30	This issue incorporates the following change: Added description about calling an API in a custom group with an IP address.
2023-04-30	This issue incorporates the following changes: ● Updated the *User Guide*. ● Changed "app" to "credential" and "VPC channel" to "load balance channel" in the *Service Overview* and *FAQs*. For details about the differences between the old and new consoles, see **Comparing Versions**.

Released On	Description
2023-03-30	This issue incorporates the following changes: ● Added **Specifications** ● Changed the name of chapter **Overview** to **Using APIG**.
2022-10-30	This issue incorporates the following changes: ● Added **Service Overview**. ● Added **Permissions Management**. ● Added **Common Errors Related to IAM Authentication Information**, **Can Applications Deployed in a VPC Call APIs?**, **Why Can't I Create a Header Parameter Named x-auth-token for an API Called Through IAM Authentication?**, and **Why Does an API Failed to Be Called Across Domains?** in the *FAQs*.
2022-07-30	This issue is the second official release. ● Added support of specifying a function alias for a FunctionGraph backend in **Creating an API**. ● Added **HTTP 2.0**.
2020-11-05	This issue is the first official release.