API Gateway

User Guide

Issue 06

Date 2024-04-03

HUAWEI

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Contents

1 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.

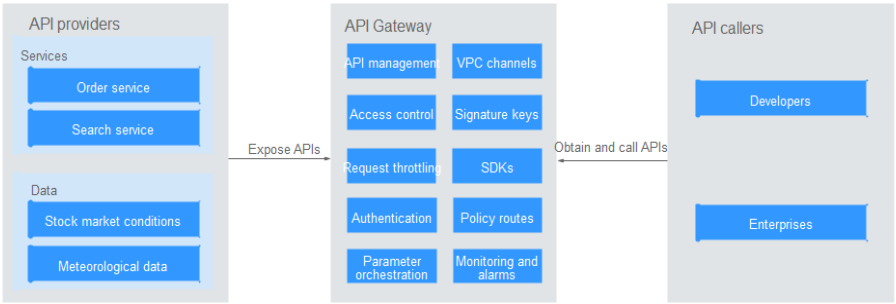
NOTICE

APIG provides **dedicated gateways** and **shared gateway** (for existing users). For details about how to use dedicated gateways, see **API Management** to **Auditing**. The shared gateway has been brought offline and can be used only by existing users. For details, see **Shared Gateway (for Existing Users)**.

General Procedure

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

Figure 1-1 APIG

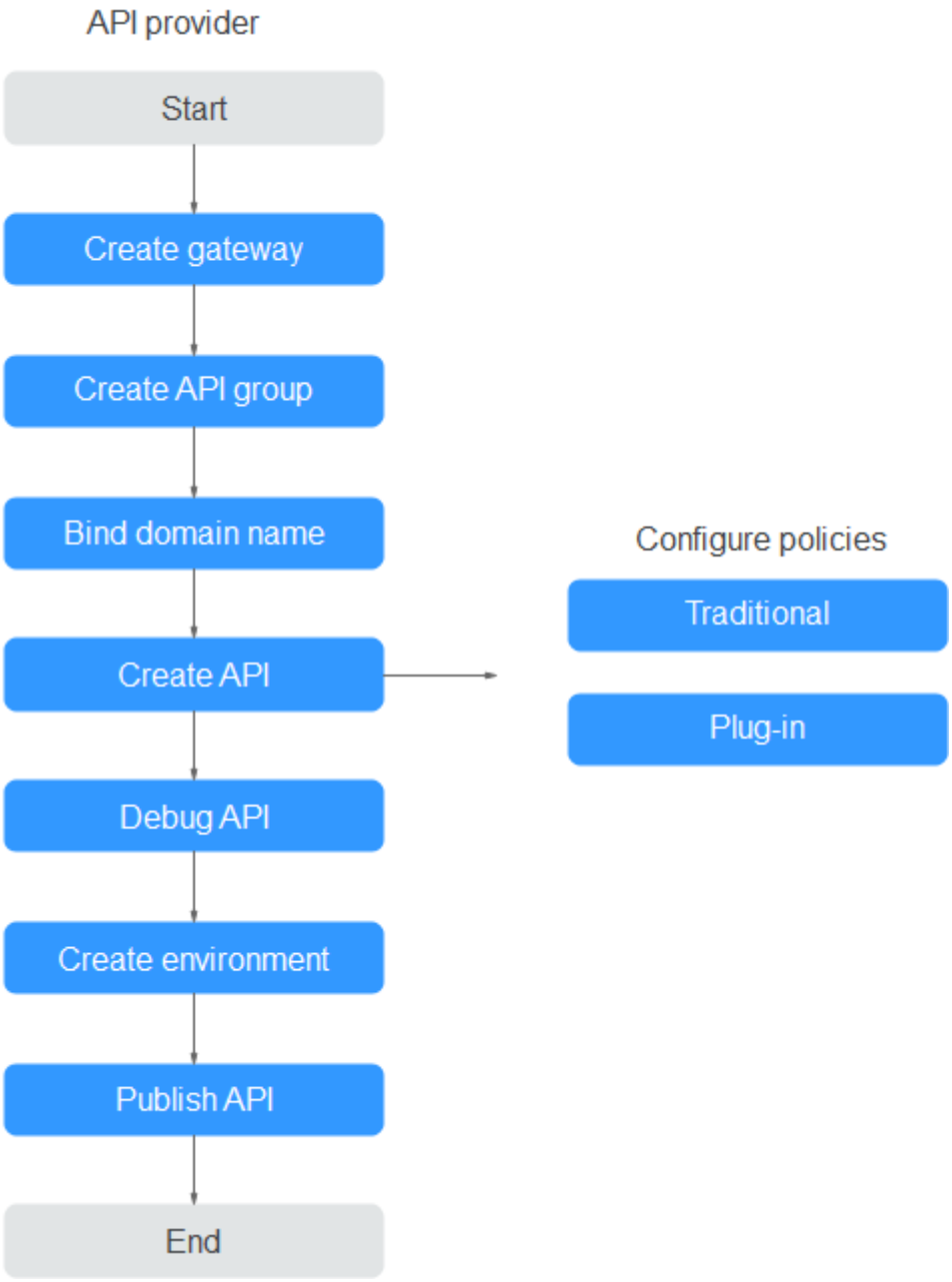


You can **expose your API services** or **obtain and call APIs of others** through APIG.

Exposing APIs

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

Figure 1-2 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.

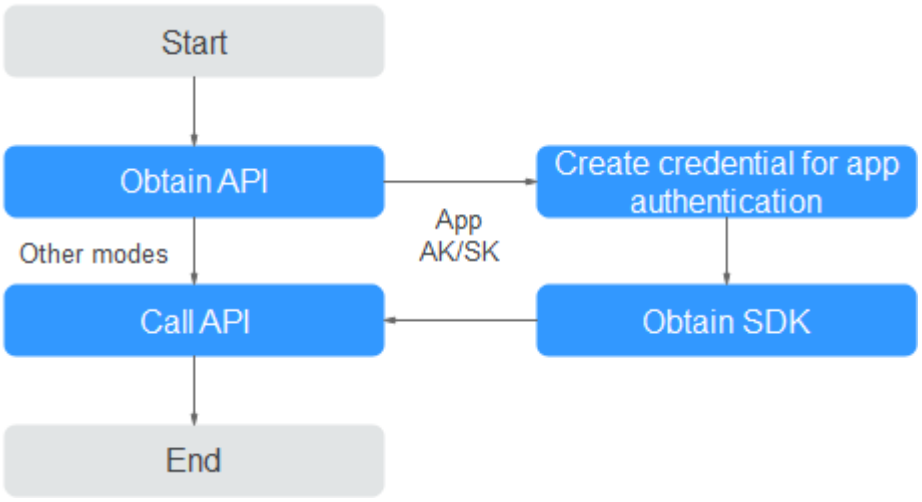
6. **Publish the API.**

   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 1-3 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.

2 API Management

2.1 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 2-1 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 2-2 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.

2.2 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 a **CCE workload**.

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 2-3 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 these workload types, see **Overview**.
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 **Labels and Annotations**.
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

Related Documents

Selectively Exposing CCE Workloads with a Dedicated Gateway

2.3 Binding a Domain Name

Before exposing APIs, bind independent domain names to the group to which the APIs belong, so that API callers can access these APIs. The APIs can also be accessed using the debugging domain name allocated to the group.

- Debugging domain name (previously called "subdomain name"): The system automatically allocates a unique debugging domain name to each API group for internal testing. The domain name can be accessed 1000 times a day, and it cannot be modified.
- Independent domain name: You can add five custom domain names for API callers to call your open APIs. There is no limit on the number of times these domain names can be accessed.

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.

Prerequisites

1. There is an independent domain name available.
2. An A record points the independent domain name to the **address** of the gateway. For details, see **Adding an A Record Set**.
3. If the API group contains HTTPS APIs, **create an SSL certificate** for the independent name.

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 2-4 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	**HTTP-to-HTTPS auto redirection** can be enabled for independent domain names.

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, client authentication (HTTPS two-way authentication) is enabled by default.
   - 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.

HTTP-to-HTTPS Auto Redirection

Gateways created after November 30, 2022 support HTTP-to-HTTPS auto redirection.

Constraints

Redirection is only suitable for GET and HEAD requests. Redirecting other requests may cause data loss due to browser restrictions.

Conditions for enabling redirection:

- The frontend request protocol is set to **HTTPS** or **HTTP&HTTPS** (see **Creating an API**).
- An independent domain name and SSL certificate have been bound to the API group to which the API belongs. For details, see the preceding descriptions in this section.

After binding an independent domain name to the API group, enable **HTTP-to-HTTPS Auto Redirection** for the 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**.

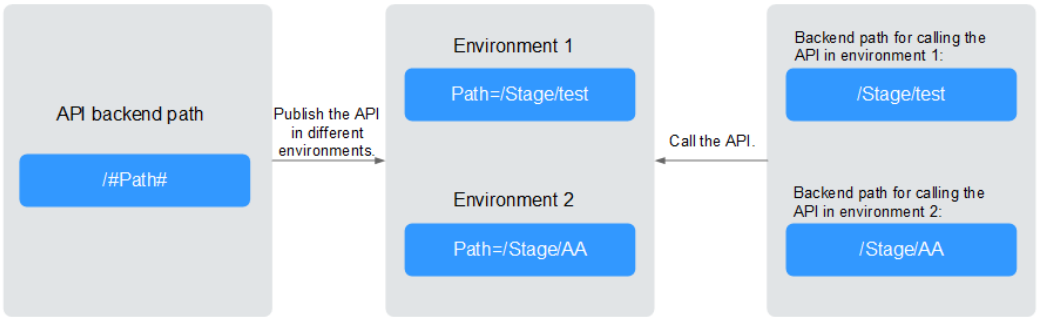
2.4 Creating an Environment Variable

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 2-1 Use of environment 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 **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 2-5 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.

2.5 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 2-6 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.
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.

Response Name	Default Status Code	Description
Unauthorized Credential	401	The credential you are using has not been authorized to call the API.

Context Variables

Table 2-7 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.
$context.error.code	Error code.
$context.error.type	Error type.

2.6 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 2-8 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.

Parameter	Description
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**: Path for requesting the API. Enclose parameters in braces. For example: **/a/{b}**. Or use a plus sign (+) to match parameters starting with specific characters. For example: **/a/{b+}**. The 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.
Matching	Options: – **Exact match**: The API can be called only using the specified request path. – **Prefix match**: The API can be called using paths starting with the matching characters. For example, if you set the request path to **/test/AA** and the matching mode to **Prefix match**, the API can be called using **/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/**. – 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.

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.
Description	Description of the API.
Content Format Type	Available for **Method** set to **POST**, **PUT**, or **ANY**. Enable to specify a content 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 content format must be supported by the backend service.
Body	Available for **Method** set to **POST**, **PUT**, **PATCH**, or **ANY**. Enter the description of the request body in the API request to help API callers understand how to correctly encapsulate API requests.

2. Configure security settings based on the following table.

Table 2-9 Security configuration

Parameter	Description
Visibility	Determine whether the API is available to the public. Options: – **Public**

Parameter	Description
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.
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 only supports HTTPS 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.

Parameter	Description
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 2-10 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.

Parameter	Description
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 2-11**, **Table 2-12**, and **Table 2-13**.

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 2-11 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#**. **NOTE** Gateways created after October 30, 2022 can transmit server name indication (SNI) to backend services during TLS handshake. ● 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: **–1**; 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 2-12 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 function version or alias. For details, see sections "Managing Versions" and "Managing Aliases" 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 2-11**. **NOTE** If the function network architecture is set to **V1**, the maximum timeout is 60,000 ms. If the network architecture is set to **V2**, the maximum timeout is 600,000 ms and can be modified by using the gateway parameter **backend_timeout**.
Backend Authentication	For details, see the description about backend authentication in **Table 2-11**.

Table 2-13 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 2-11**.
Add Header	Customize the response header parameters for the API. Click **Add Header**, and enter the parameter name, value, and description.

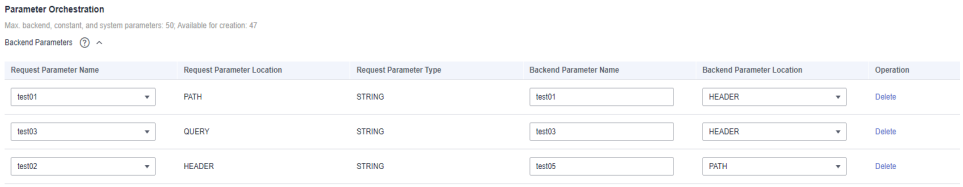
NOTE

- 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 2-2**) based on the parameters and their locations in backend requests.

Figure 2-2 Configuring backend parameters



   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 2-14 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 2-15 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 **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 2-16**. For details about other parameters, see **Table 2-11**, **Table 2-12**, and **Table 2-13**.

Table 2-16 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 2-17**.

Table 2-17 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 2-18 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 a Backend Service Fails to Be Invoked or the Invocation Times Out?

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**.

2.7 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 2-19 gRPC vs REST

Item	gRPC	REST
Message encoding	protobuf	JSON
Transmission protocol	HTTP/2	HTTP
Transmission performance	Fast, with less content to transmit	More content to transmit
Transmission mode	• Unary RPC Send a single request and receive a single response. • Server streaming RPC Send a single request and receive a single response. • 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 server.

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}*.
- Base64 encoding is not supported.

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.

  NOTE

  gRPC APIs with a FunctionGraph backend are supported in CN Southwest-Guiyang1, CN East-Shanghai1, CN North-Beijing4, CN East-Shanghai2, and LA-Santiago.

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

Related Documents

Routing gRPC Service Requests

2.8 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

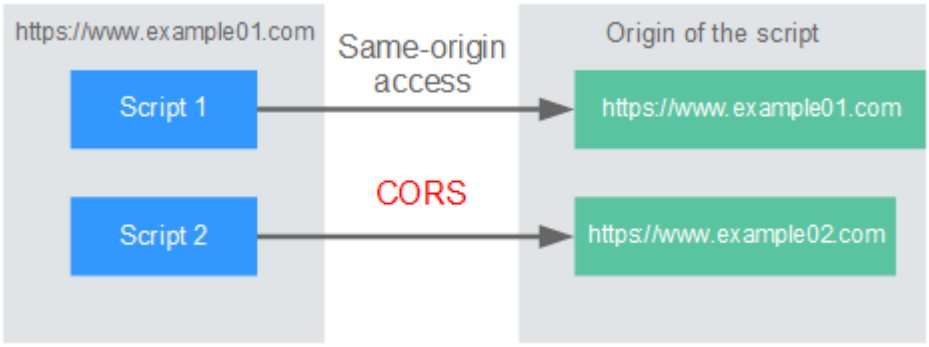
After cloning an API, verify it by following the procedure in **Debugging an API**.

2.9 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 2-3 Process flow of the CORS mechanism



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.

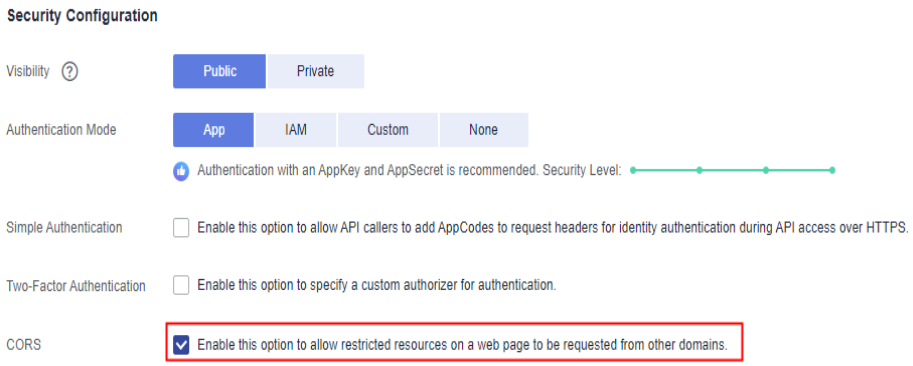
  Before sending a not-so-simple request, browsers send an HTTP preflight request to the target server to confirm whether the origin the web page is loaded from is in the allowed origin list, and to confirm which HTTP request methods and header fields can be used. If the preflight request is successful,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



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

NOTICE

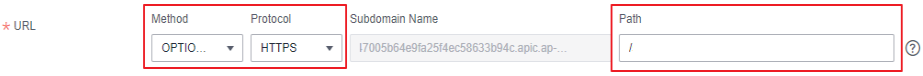
If your API will receive not-so-simple requests, **create another API that will be accessed using the OPTIONS method** in the same group as the target API to receive preflight requests.

Follow this procedure to define the preflight request API. For more information, see **Not-So-Simple Request**.

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

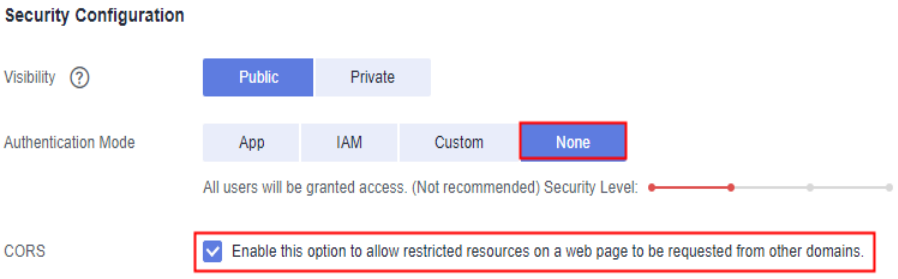
- **Method**: Select **OPTIONS**.
- **Protocol**: The same protocol used by the API with CORS enabled.
- **Path**: Enter a slash (/).

Figure 2-4 Defining the API request



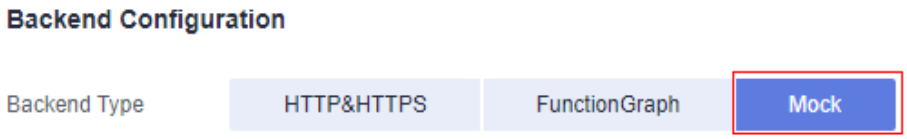
b. In the **Security Configuration** area, select **None** and enable **CORS**.

Figure 2-5 None authentication



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

Figure 2-6 Mock backend service



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.

NOTE

If you use the CORS plug-in policy for an API, you do not need to create another API that uses 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"}
```

2.10 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 an API has been bound with a request throttling policy, the policy will not work during debugging of the API.
- 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.

2.11 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.

2.12 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?

Why Can't APIs Published in a Non-RELEASE Environment Be Accessed?

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

2.13 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.

2.14 Importing and Exorting APIs

2.14.1 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 2-20 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 2-21 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 2-22 Differences in API request path template syntax

Syntax	OASSwagger	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

2.14.2 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.

2.14.3 Exporting APIs

You can export APIs one by one or in batches as JSON or YAML 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 2-23 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.
API	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	Export APIs in **JSON** or **YAML** format.
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

2.14.4 Extended Definition

2.14.4.1 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 2-24 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.

2.14.4.2 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 2-25 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.

2.14.4.3 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 2-26 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.

2.14.4.4 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 2-27 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.

2.14.4.5 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 2-28 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

2.14.4.6 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 2-29 Parameter description

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

2.14.4.7 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 2-30 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.

2.14.4.8 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 2-31 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.

2.14.4.9 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 2-32 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**.

2.14.4.10 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 2-33 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**.

2.14.4.11 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 2-34 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**.

2.14.4.12 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 2-35 Parameter description

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

2.14.4.13 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 2-36 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.

2.14.4.14 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 2-37 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**.

2.14.4.15 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 2-38 Parameter description

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

2.14.4.16 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 2-39 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.

2.14.4.17 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 2-40 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.

2.14.4.18 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 2-41 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.

2.14.4.19 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 2-42 Parameter description

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

2.14.4.20 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 2-43 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.

2.14.4.21 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 2-44 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 (,).

2.14.4.22 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 2-45 Parameter description

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

2.15 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

2.16 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.

3 API Policies

3.1 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 3-1 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**: Authenticate 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 an API group and then select 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

3.2 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 3-2 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
}
```

3.3 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 cannot modify the response headers (including **x-apig-*** and **x-request-id**) added by APIG or the headers required for CORS.

Configuration Parameters

Table 3-3 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"
    }
  ]
}
```

3.4 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

- A traditional request throttling policy becomes invalid if a request throttling 2.0 policy is bound to the same API as the traditional one.
- You can define a maximum of 100 parameter-based throttling rules. The parameter name is max. 32 characters.
- The policy content cannot exceed 65,535 characters.

Parameter Description

Table 3-4 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. ● 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 Name**: 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**. • Conditions 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 **Obtaining a Project ID**. • Specify an account ID (not IAM user ID) for an API with IAM authentication. For details, see **Obtaining an Account Name and Account ID**. **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
  }
 ]
}
```

3.5 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

- 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 3-5 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
Message Tx/Rx Mechanism	Message transmission and receiving mechanism of the target Kafka. The default value is **PLAIN**.

Parameter	Description
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.
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.

Parameter	Description
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 (,).

3.6 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.

Parameter Description

Table 3-6 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.
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.

Parameter	Description
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.
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**.

Parameter	Description
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, 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**: version 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
  }
}]
}
```

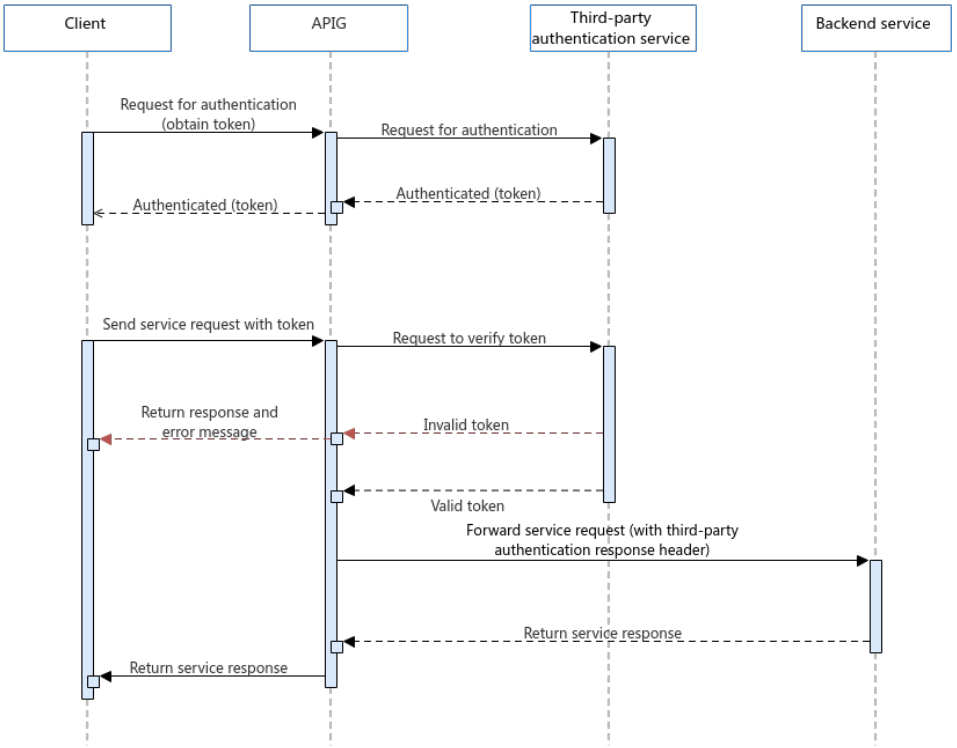

3.7 Third-Party Authorizer

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**.



Configuration Parameters

Table 3-7 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.
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.
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.
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"
}
```


3.8 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

- 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 3-8 Parameter description

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	The maximum number of times each bound API can be called by an IP address within the specified period. • 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 app 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 Apps** tab.

Step 2 Click **Select Excluded App**.

Step 3 Select an app 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 app 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 apps (max. **2** API requests for app A and max. **4** API requests for app B). If the request throttling policy is bound to an API, apps 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 3-9 Excluded tenant configuration

Parameter	Description
Tenant ID	Account ID or project ID. For details, see the description about **Excluded Tenants** in Table 3-4.
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

3.9 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

- 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.
- Gateways created after December 31, 2022 support API access control by **account ID**. If you need to use this function on dedicated gateways created earlier, contact customer service.

Configuration Parameters

Table 3-10 Parameter description

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. • **Account ID**: Control IAM authentication–based API access by account ID, not IAM user ID. **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.
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 Address	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 Name	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.

Parameter	Description
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.

3.10 Signature Keys

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

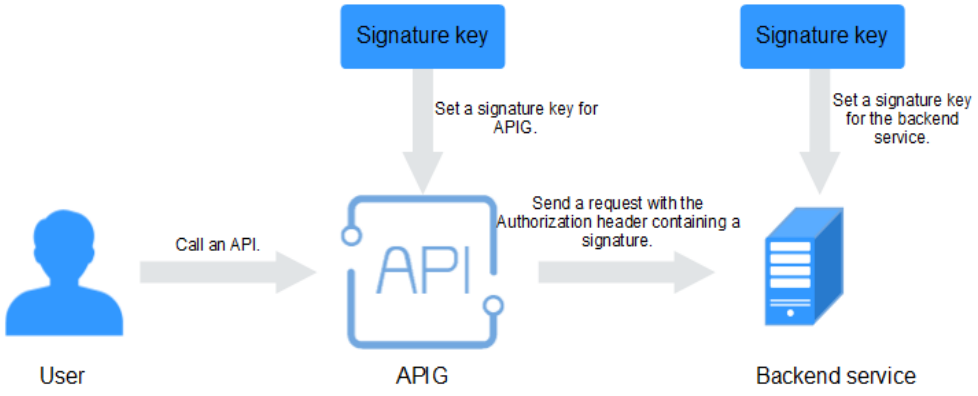
A signature key consists of a key and secret, and can be used only after being bound to an API. When an API bound with a signature key is called, APIG adds signature details to the API request. The backend service of the API signs the request in the same way, and verifies the identity of APIG by checking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

Usage Guidelines

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 3-1 Signature key process flow



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 3-11 Parameter description

Parameter	Description
Name	Signature key name.
Type	Authentication type. Options: **HMAC**, **Basic auth**, **AES**, and **Public key**. **Public key** is available only if **public_key_enable** has been turned on on the **Parameters page** of the gateway.
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. • If **Type** is **Public key**, enter the public key used for authentication.
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. • If **Type** is **Public key**, enter the private key used for authentication.
Confirm Secret	Enter the secret again.

Verifying the Signing Result

Sign each backend request by following the instructions in **Signature Algorithm**, and check whether the backend signature is consistent with the signature in the **Authorization** header of the API request.

3.11 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.

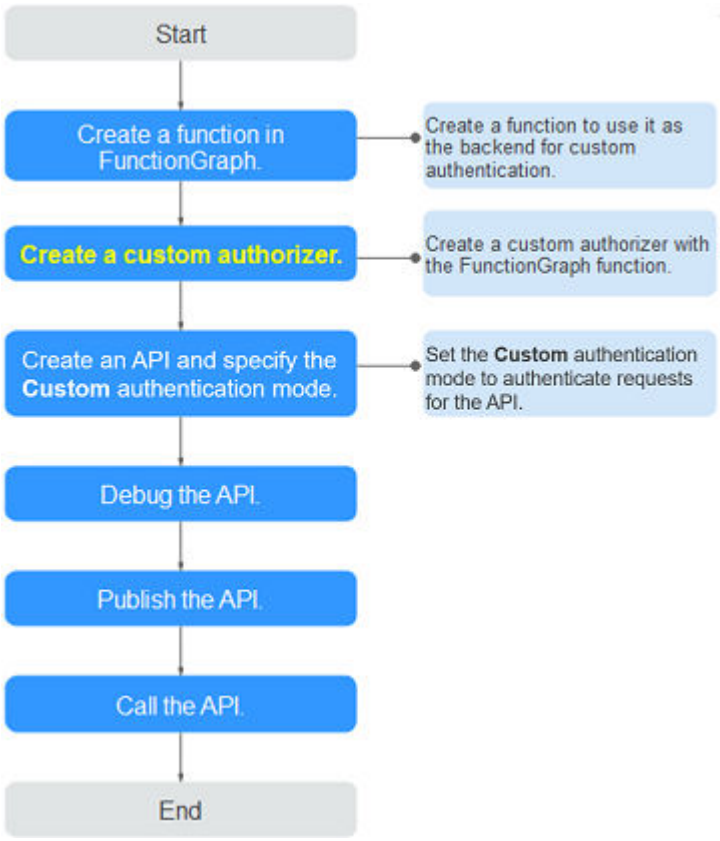
NOTE

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

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

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

Figure 3-2 Calling APIs through custom authentication



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 3-12 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.
Version/Alias	Select a function version or alias. For details, see **FunctionGraph User Guide**.
Max. Cache Age (s)	The time for caching authentication results. Value **0** means that authentication results will not be cached. The maximum value is **3600**.
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.

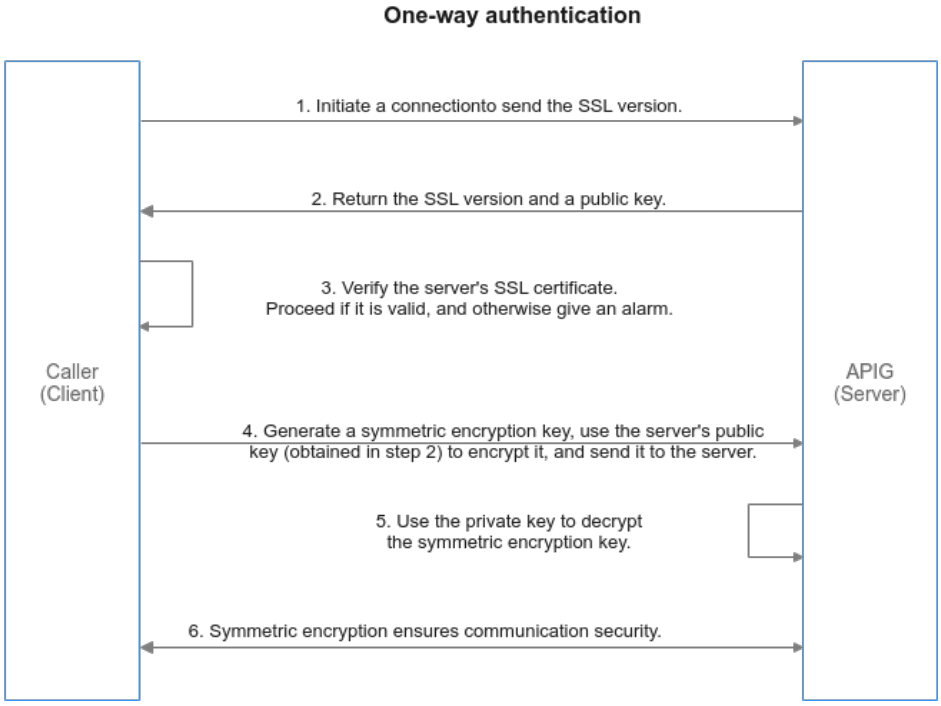
Step 5 Click **OK**.

----End

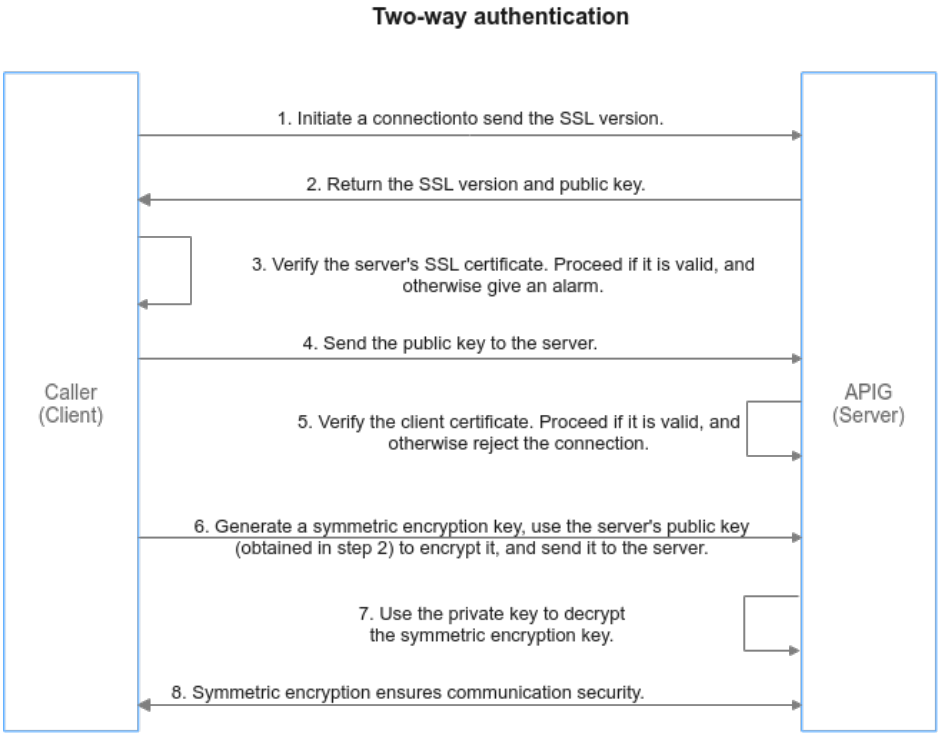
3.12 SSL Certificates

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, and support both one-way and two-way authentication.

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



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



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 3-13 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**.

Parameter	Description
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**.

Format	Converting with OpenSSL
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

To update an SSL certificate, go to the certificate list page, click **Modify** in the **Operation** column of the target certificate, and then modify it.

If the updated SSL certificate has been bound to an independent domain name, the client authentication (HTTPS two-way authentication) is enabled 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.

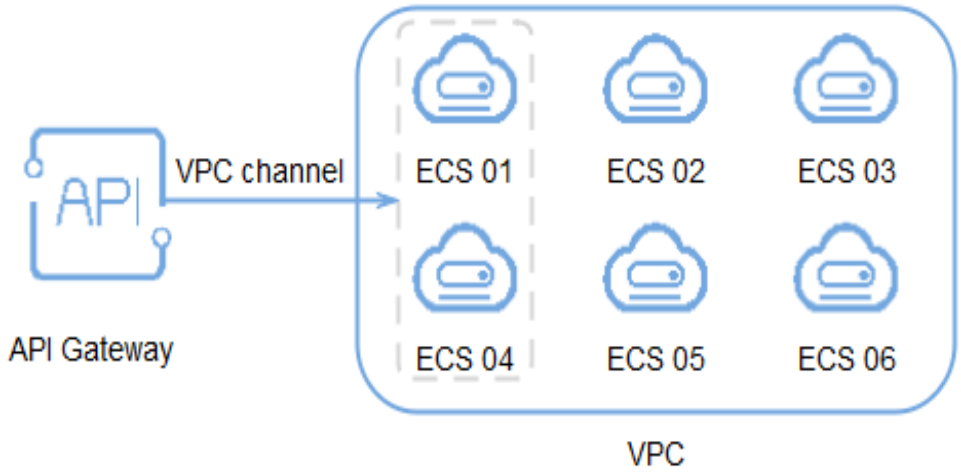
3.13 Load Balance Channels

Load balance channels expose your services through **dedicated gateways**. and are accessed through subnets in VPCs for lower latency. They balance the loads of backend services as server channels or automatically synchronize service node changes as microservice channels.

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 3-3 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 microservice channel, ensure that you have **created a cluster** (a CCE cluster of VPC network model or a Turbo cluster) and **a workload**

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 **Pod Labels and Annotations**.
  - 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 3-14 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
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.

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 3-15 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 3-15**.

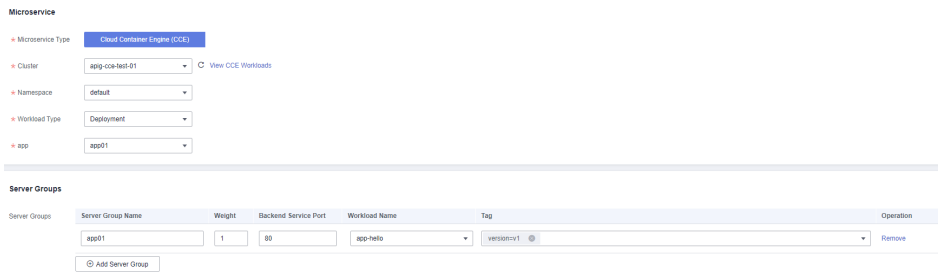
b. Click **Add Backend Server Address** and enter a backend server address.

Table 3-16 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.
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 a microservice and a server group.



1. Configure microservice information according to the following table.

Table 3-17 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.
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 these workload types, see **Overview**.
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 **Labels and Annotations**.
Service Label Value	

2. Configure a server group.

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

Table 3-18 Server group parameters

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	By default, the port of the load balance channel is used.

Parameter	Description
Workload Name	Select a CCE workload.

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 3-19 Basic information

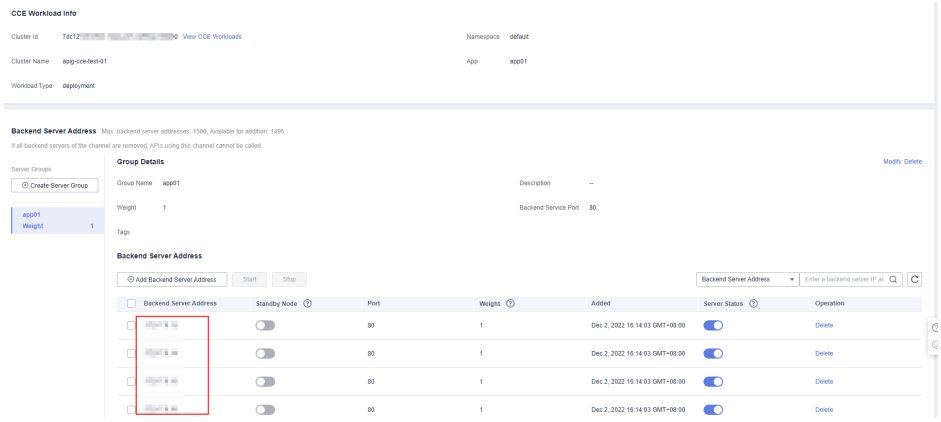
Parameter	Description
Protocol	The protocol used to perform health checks on cloud servers associated with the channel. Options: • TCP • HTTP • HTTPS Default value: **TCP**.
Two-Way Authentication	**Set this parameter only when Protocol is set to HTTPS.** Determine whether to allow APIG to authenticate 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.

Figure 3-4 Microservice load balance channel details



----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.

Related Documents

Selectively Exposing CCE Workloads

3.14 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 3-20 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.

4 Credentials

4.1 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 **Developer Guide**.

NOTE

- APIs that use IAM 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 4-1 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

4.2 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

4.3 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.
- 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 4-2 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

4.4 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 4-3**.

Table 4-3 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.
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

4.5 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 4-4 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

5 Monitoring & Analysis

5.1 API Monitoring

5.1.1 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 5-1 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 5-2 Monitoring dimensions

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

5.1.2 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 **Creating an Alarm Rule**.

----End

5.1.3 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.

Step 7 Click **More** to view more metrics on the Cloud Eye console.

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 target API group and view its metrics.

----End

5.2 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 5-3 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.
Period	A cycle when data is aggregated to calculate the maximum, minimum, average, total, or variance value.

----End

5.3 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 **Log Management**.

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 **Log Transfer**.

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

Table 5-4 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.
6	time_local	Time when a request is received.
7	request_time	Request latency.
8	request_method	HTTP request method.
9	scheme	Request protocol.

No.	Field	Description
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.
28	access_model1	Authentication mode 1.
29	access_model2	Authentication mode 2. Enabling two-factor authentication will use the custom authorizer ID.

No.	Field	Description
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.
52	group_id	Group ID.
53	apig_err_code	Gateway error code.

----End

6 Gateway Management

6.1 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 on Buying a Gateway

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. A platinum *X* requires 4 more private IP addresses than the previous edition. Check that the subnet you choose has sufficient private IP addresses on the VPC console.

Network Environment

- Workload

  Gateways are deployed in VPCs. 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 **Buy Gateway page**.

NOTE

- ELB load balancing is enabled by default after gateways are purchased in regions except **LA-Mexico City1** and **CN North-Beijing1**. 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 Set the gateway parameters by referring to the following table.

Table 6-1 API gateway parameters

Parameter	Description
Billing Mode	Billing mode of the dedicated gateway. Options: **Pay-per-use**.

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*. **NOTE** Currently, platinum edition 2 and later are available only in **CN-Hong Kong**.
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.
Enterprise Project	Select an enterprise project to which the gateway belongs. This parameter is available only if your account is an enterprise account. For details about resource usage, migration, and user permissions of enterprise projects, see the **Enterprise Management User Guide**.

Parameter	Description
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. You will need to **pay** for the EIP usage. **NOTE** • Gateways with public inbound access enabled in **CN-Hong Kong**, **CN East-Shanghai1**, and **CN Southwest-Guiyang1** are randomly assigned an EIP and cannot use an existing EIP. Set a bandwidth that meets your service requirements for public inbound access. The bandwidth will be billed by hour based on the pricing of the EIP service. • 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.
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. Cloud resources (such as ECSs) within the same VPC can call APIs using the private IP address of the gateway. Deploy the gateway in the same VPC as your other services to facilitate network configuration and secure network access.
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).

Parameter	Description
Advanced Settings	Enable this option to set tags. Alternatively, set tags on the Tag Management Service (TMS) console by referring to **Tag Management**.
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** to create one on the TMS console. **NOTE** If your organization has configured tag policies for APIG, add tags to gateways based on the policies. If a tag does not comply with the policies, gateway creation may fail. Contact your organization administrator to learn more about tag policies.
Description	Description about the gateway.

Step 3 Click **Next**.

Step 4 Confirm the gateway configurations, read and confirm your acceptance of the service agreement, and click **Pay Now**. 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.

6.2 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 6-2 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	Billing mode of the gateway.
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.
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**.

Modifiable Parameter	Description
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

6.3 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 6-3 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 body size allowed for an API request. Default: 12 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_api_key	Determine whether to enable app_api_key authentication. Default: disabled. If you enable this function, the **apikey** parameter can be added to the API request to carry the key of a credential for authentication.
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.

Parameter	Description
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.
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.
app_jwt_enable	Determine whether to enable app_jwt authentication. Default: disabled. If you enable this function, the **Authorization** and **Timestamp** parameters can be added to API requests to carry the key, secret, and timestamp of a credential for authentication. **app_jwt_auth_header** is a header included in API requests for app_jwt authentication. The default value of the header is **Authorization**.

Parameter	Description
public_key_enable	Determine whether to enable public_key authentication. Default: disabled. If you enable this option, signature keys of the public_key type can be used for authentication. **public_key_uri_prefix** indicates the prefix of the URI used to obtain the secret of public_key. The URI format is as follows: **https://***{VPC access address}{public_key_uri_prefix}{public_key name}*.
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.
request_custom_config	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 6-4**.

Table 6-4 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

6.4 Tag Management

Tags classify your gateways to facilitate search, analysis, and management.

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.

NOTE

If your organization has configured tag policies for APIG, add tags to gateways based on the policies. If a tag does not comply with the policies, tag addition may fail. Contact your organization administrator to learn more about tag policies.

Step 5 Click **OK**.

----End

6.5 Managing VPC Endpoints

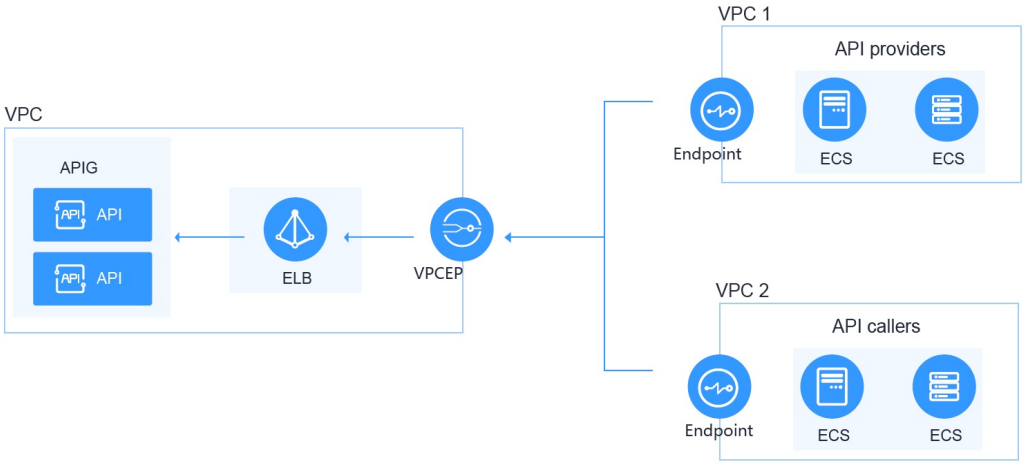
VPC endpoints are secure and private channels for connecting VPCs to VPC endpoint services.

APIs can be exposed and accessed across VPCs in the same region of the same cloud.

NOTICE

Currently, regions except **LA-Mexico City1** and **CN North-Beijing1** support the VPC endpoint management.

Figure 6-1 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 **VPC Endpoints**.

Table 6-5 VPC endpoint information

Parameter	Description
VPC Endpoint Service	Name of the VPC endpoint service created when you buy the gateway. The gateway can be accessed using the endpoint service.

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 **What Are Statuses of VPC Endpoint Services and VPC Endpoints?** • **Owner**: **account ID** 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**. • **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

6.6 Modifying Specifications

If the specifications of a gateway cannot meet your service requirements, upgrade the specifications.

NOTICE

- Currently, the specifications of ELB-based gateways can be modified in regions except **LA-Mexico City1** and **CN North-Beijing1**. 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 6-3**.

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

7 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

8 Published API Calling

8.1 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 **VPC Peering Connection**.
- 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 **Connecting VPCs in Different Regions**.

- 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_api_key)	Obtain the key and secret of a credential authorized for the API from the API provider.
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.
App (app_jwt)	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.

Authentication Mode	Authentication Information
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**. You can also access the API using an IP address.

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. • 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 **Calling APIs Through App Authentication**.
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**.
App (app_api_key)	• To enable app_api_key authentication, ensure that the **app_api_key** parameter has been set to **on** on the **Parameters** tab of the gateway. • Add the header or query string **apikey** and set the parameter value to the key obtained in **Obtaining API Calling Information**.

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, and set **app_api_key** to **off** to disable app_api_key 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 (app_jwt)	• To enable app_jwt authentication, ensure that the **app_jwt** parameter has been set to **on** on the **Parameters** tab of the gateway. • Add the header parameter **Timestamp** and set the parameter value to the Unix timestamp of the current time in millisecond. • Add the header parameter **Authorization** and set the parameter value to "SHA-256 (*appkey* + *appsecret* + *timestamp*)", in which *appkey* and *appsecret* are the key and secret obtained in **Obtaining API Calling Information** and *timestamp* is the Unix timestamp of the current time in millisecond. The character string encrypted using SHA-256 must be lowercase letters. • Add the header parameter **X-HW-ID** and set the parameter value to the key 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 **Token Authentication**.
IAM (AK/SK)	Use the obtained SDK to sign the API request. For details, see **AK/SK Authentication**.

Authentication Mode	Request Parameter Configuration
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.

8.2 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.

Response Header	Description	Remarks
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**.
X-Apig-count	Total number of times that a request is forwarded by APIG.	Returned for all valid requests forwarded by APIG. If the value of **X-Apig-count** is greater than 10, error APIG.0612 is reported.

8.3 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 **Error Codes**.

NOTE

- For details about the error codes that may occur when you manage APIs, see **Error Codes**.
- 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 8-1 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.
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.

Error Code	Error Message	HTTP Status Code	Description	Solution
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.
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.

Error Code	Error Message	HTTP Status Code	Description	Solution
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.
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.

Error Code	Error Message	HTTP Status Code	Description	Solution
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 **Calling APIs Through App Authentication**.
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.

9 Permissions Management

9.1 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 Huawei Cloud account does not require individual IAM users, skip this chapter.

This section describes the procedure for granting permissions (see **Figure 9-1**).

Prerequisites

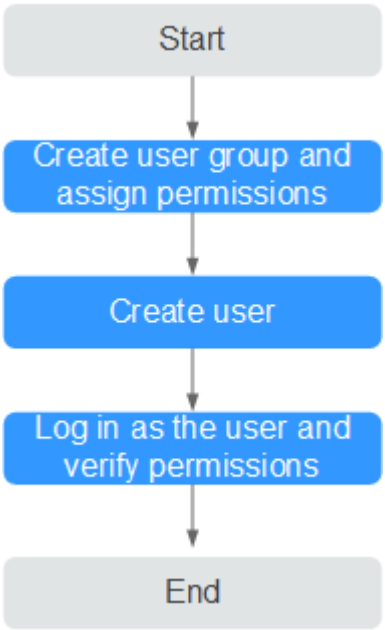
Learn about the permissions (see **Table 9-1**) supported by APIG and choose policies or roles according to your requirements. For the permissions of other services, see **System Permissions**.

Table 9-1 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 9-1 Process for granting APIG permissions



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.

9.2 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 **Permissions Policies and Supported Actions**.

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 **Creating a Custom Policy**. 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
             "
        ]
    }
  ]
}
```


10 Auditing

10.1 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)**.

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 10-1 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
Creating an environment	Environment	createEnvironment

Operation	Resource Type	Trace Name
Modifying an environment	Environment	updateEnvironment
Deleting an environment	Environment	deleteEnvironment
Creating a variable	EnvVariable	CreateEnvironmentVariable
Deleting a variable	EnvVariable	DeleteEnvironmentVariable
Modifying a variable	EnvVariable	UpdateEnvironmentVariable
Creating a request throttling policy	Throttle	CreateRequestThrottlingPolicy
Modifying a request throttling policy	Throttle	UpdateRequestThrottlingPolicy
Deleting a request throttling policy	Throttle	DeleteRequestThrottlingPolicy
Creating an API	Api	CreateApi
Modifying an API	Api	UpdateApi
Deleting an API	Api	DeleteApi
Publishing an API or taking an API offline	Api	CreateOrDeletePublishRecordForApi
Verifying the API definition	Api	CheckApis
Debugging an API	Api	DebugApi
Publishing multiple APIs or taking APIs offline	Api	BatchPublishOrOfflineApi
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
Unbinding a signature key	SignatureBinding	DisassociateSignatureKey

Operation	Resource Type	Trace Name
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
Binding an access control policy to an API	AclBinding	CreateApiAclBinding

Operation	Resource Type	Trace Name
Unbinding an access control policy from an API	AclBinding	DeleteApiAclBinding
Unbinding multiple access control policies from APIs	AclBinding	BatchDeleteApiAclBinding
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
Updating a Backend Server Group of a VPC Channel	Vpc	UpdateMemberGroup

Operation	Resource Type	Trace Name
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	UpdateGatewayResponseType
Deleting the response of an error type defined for an API group	ApiGroup	DeleteGatewayResponseType
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
Disabling public inbound access for a gateway	Instance	RemoveIngressEip

Operation	Resource Type	Trace Name
Deleting a dedicated gateway	Instance	DeleteInstances
Modifying the specifications of a pay-per-use gateway	Instance	CreatePostPayResizeOrder
Accepting or rejecting a VPC endpoint connection	vpc-endpoint	AcceptOrRejectEndpointConnections
Adding whitelist records for a VPC endpoint service	vpc-endpoint	AddEndpointPermissions
Deleting whitelist records of a VPC endpoint service	vpc-endpoint	DeleteEndpointPermissions
Batch adding or deleting gateway tags	Instance	BatchCreateOrDeleteInstanceTags
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	BatchDisassociateDomains
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
Binding a plug-in to an API	Plugin	AttachPluginToApi

Operation	Resource Type	Trace Name
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 **Deleting a Tracker**.

10.2 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**

Constraints

- Traces of a single account can be viewed on the CTS console. Multi-account traces can be viewed only on the **Trace List** page of each account, or in the OBS bucket or the **CTS/system** log stream configured for the management tracker with the organization function enabled.
- You can only query operation records of the last seven days on the CTS console. To store operation records for more than seven days, you must configure an OBS bucket to transfer records to it. Otherwise, you cannot query the operation records generated seven days ago.
- After performing operations on the cloud, you can query management traces on the CTS console 1 minute later and query data traces on the CTS console 5 minutes later.

Viewing Real-Time Traces in the Trace List of the New Edition

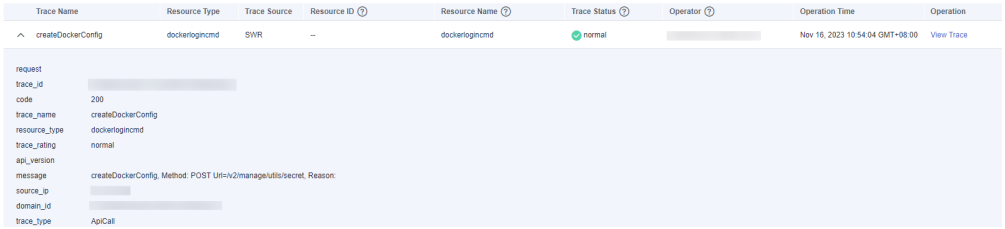
1. Log in to the management console.
2. Click  in the upper left corner and choose **Management & GovernanceManagement & Deployment** > **Cloud Trace Service**. The CTS console is displayed.
3. Choose **Trace List** in the navigation pane on the left.
4. On the **Trace List** page, use advanced search 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 click 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 **Trace Structure**section "Trace References" > "Trace Structure" and **Example Traces**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.

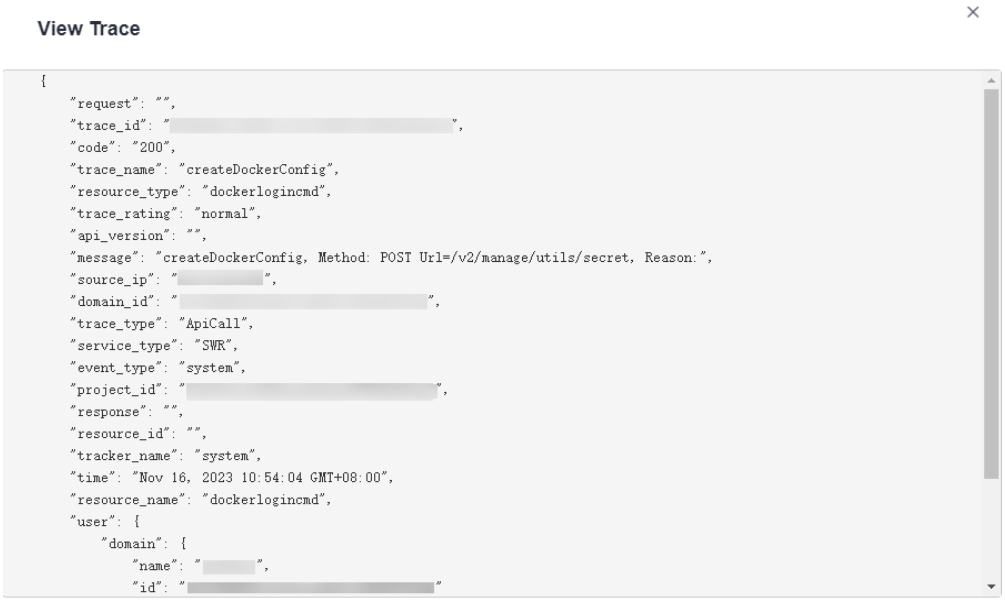
Viewing Real-Time Traces in the Trace List of the Old Edition

1. Log in to the management console.
2. Click in the upper left corner and choose **Management & GovernanceManagement & 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.



10. For details about key fields in the trace structure, see **Trace Structure**section "Trace References" > "Trace Structure" and **Example Traces**section "Trace References" > "Example Traces".
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.

11 Shared Gateway (for Existing Users)

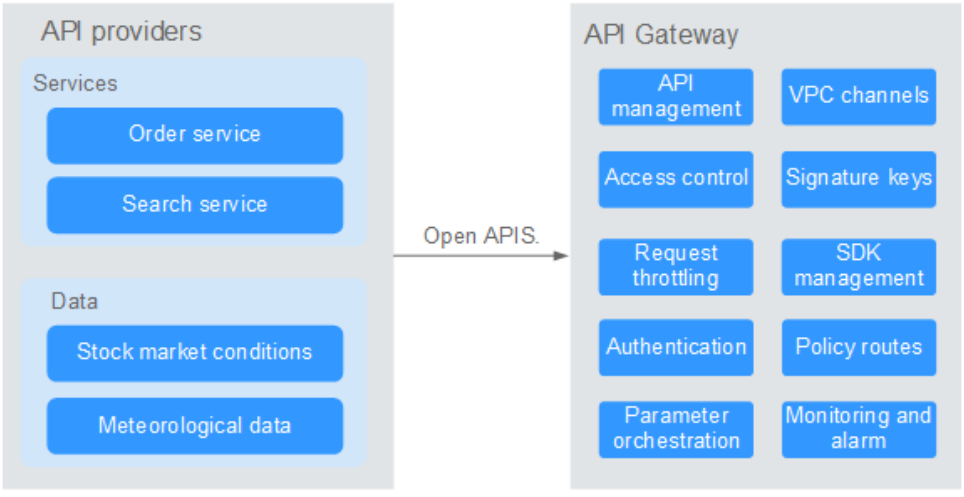
11.1 Using APIG

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 through its **API opening** and **API calling** functions.

- **API Opening**

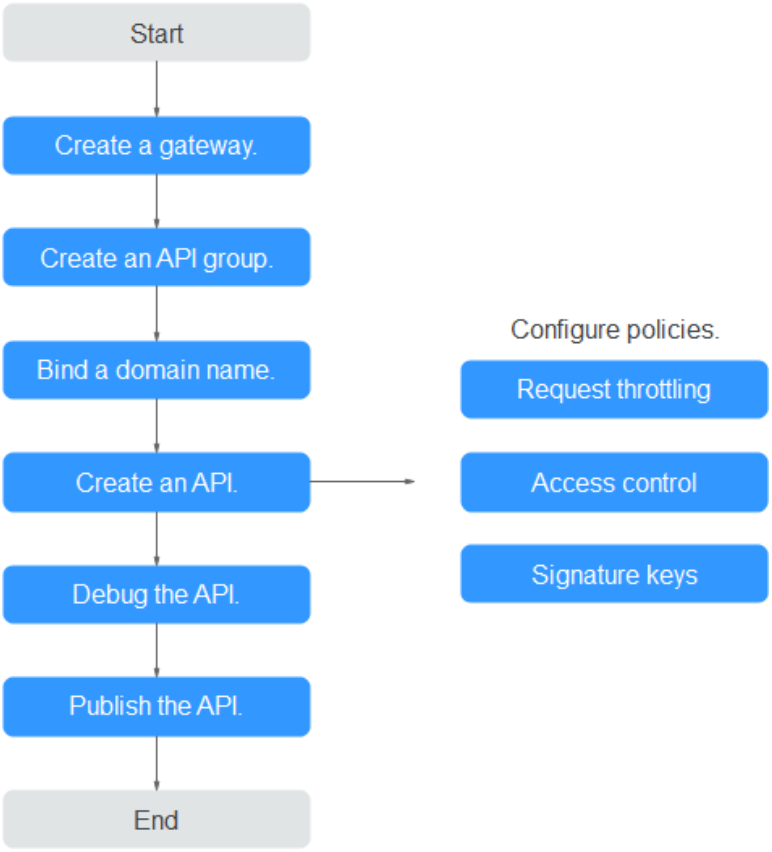
  Enterprises and developers selectively expose their services and data through APIG.

  Figure 11-1 API opening



The following figure shows the API opening process.

Figure 11-2 API opening process



a. Create a gateway.

   Alternatively, use the **shared gateway**.

b. **Create an API group**.

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

c. **Bind a domain name**.

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

   You can debug the API using the default subdomain name allocated to the group to which the API belongs. The subdomain name can be called a maximum of 1000 times every day.

d. **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:

   - **Request throttling**

     Set the maximum number of times the 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 and ensure secure access.

e. **Debug the API**.

Verify whether the API is working normally.

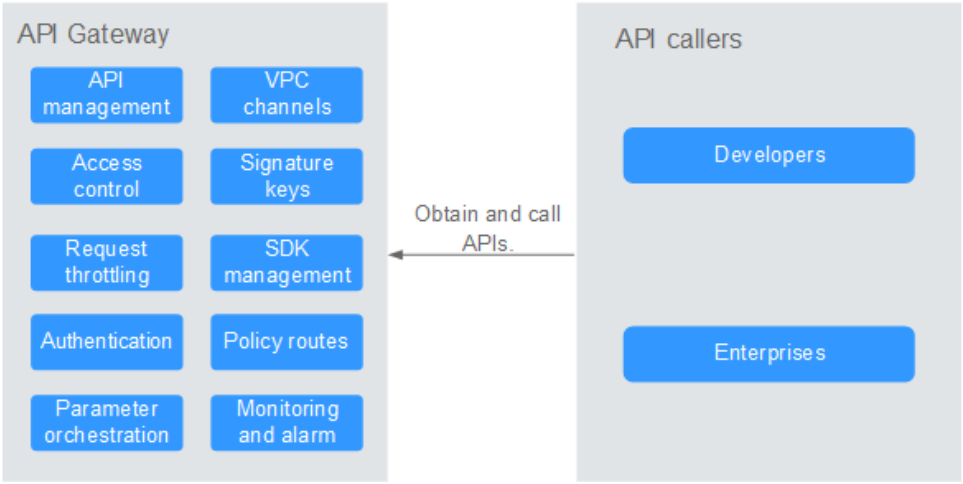
f. **Publish the API**.

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

- **API calling**

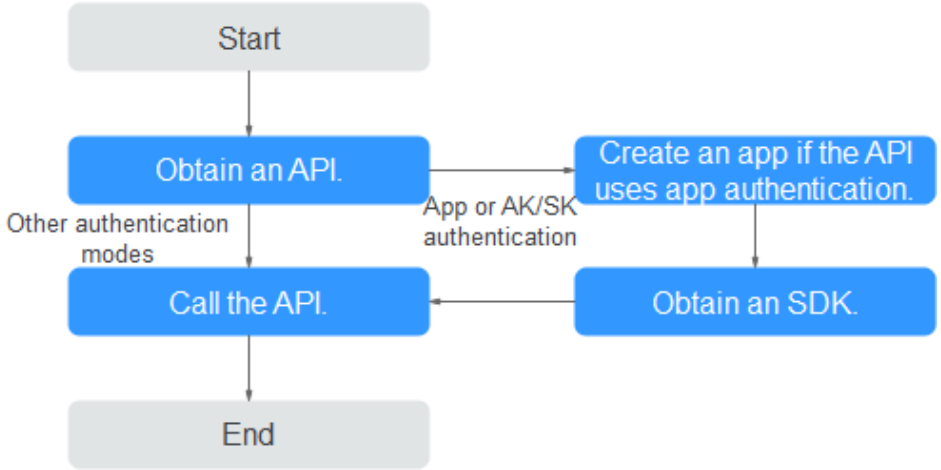
Enterprises and developers obtain and call APIs of other providers, thereby reducing development time and costs.

Figure 11-3 API calling



The following figure shows the API calling process.

Figure 11-4 API calling process



a. **Obtain an API**.

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

b. **Create an app**.

For an API that uses app authentication, create an app to generate an AppKey and AppSecret. Bind the app to the API so that you can call the API through app authentication.

c. **Obtain an SDK**.

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

d. **Call the API**.

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

11.2 Accessing the Shared Gateway

To access the shared gateway console as an existing user, perform the following steps:

Step 1 Log in to the **APIG console**.

Step 2 In the upper right corner of the **Overview** page, click **Access Shared Gateway**. The **Shared Gateway** page is displayed.

----End

11.3 API Group Management

11.3.1 Creating an API Group

Scenario

Before creating an API, you must create an API group. An API group contains different APIs used for the same service.

NOTE

Each API can only belong to one API group.

Procedure

Step 1 **Access the shared gateway console**.

Step 2 In the navigation pane, choose **API Publishing** > **API Groups**.

Step 3 Click **Create API Group**, and set the parameters described in **Table 11-1**.

Table 11-1 Parameters for creating an API group

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

Step 4 Click **OK**.

After the API group is created, it is displayed in the API group list.

NOTE

- The system automatically allocates a subdomain name to the API group for internal testing. The subdomain name can be accessed 1000 times a day.
- APIs created in the shared gateway can be accessed over public networks by using the subdomain name of the group to which the APIs belong.
- To make your APIs available for users to access, bind independent domain names to the API group to which the APIs belong.

----End

Follow-Up Operations

After the API group is created, bind independent domain names to it so that API callers can use the domain names to call APIs in the group. For more information, see **Binding a Domain Name**.

11.3.2 Binding a Domain Name

Scenario

Before you open an API, you must bind one or more independent domain names to the group to which the API belongs.

NOTE

- In the shared gateway, you cannot bind the same independent domain name to different API groups.

Note the following points before you bind a domain name:

- Subdomain name: After an API group is created, the system automatically allocates a unique subdomain name to it for internal testing. The subdomain name can be accessed 1000 times a day, but it cannot be modified.
- Independent domain name: You can add five custom domain names for API callers to call your open APIs. There is no limit on the number of times these domain names can be accessed.

Prerequisites

1. There is an independent domain name available.
2. Shared gateway: A CNAME record points the independent domain name to the subdomain name of the API group. For details, see **Adding a CNAME Record Set**.
3. If the API group contains APIs that are called through HTTPS, there needs to be **SSL certificates** configured for the independent domain name. SSL certificates can only be added manually with a custom name, content, and a key.

Procedure

Step 1 Access the shared gateway console.

Step 2 In the navigation pane, choose **API Publishing** > **API Groups**.

Step 3 Go to the **Domain Names** tab page using one of the following methods:

- Click the name of the target API group, and click the **Domain Names** tab on the displayed API group details page.
- In the **Operation** column of the target API group, choose **More** > **Manage Domain Name**.

Step 4 Click **Bind Independent Domain Name** and enter a domain name.

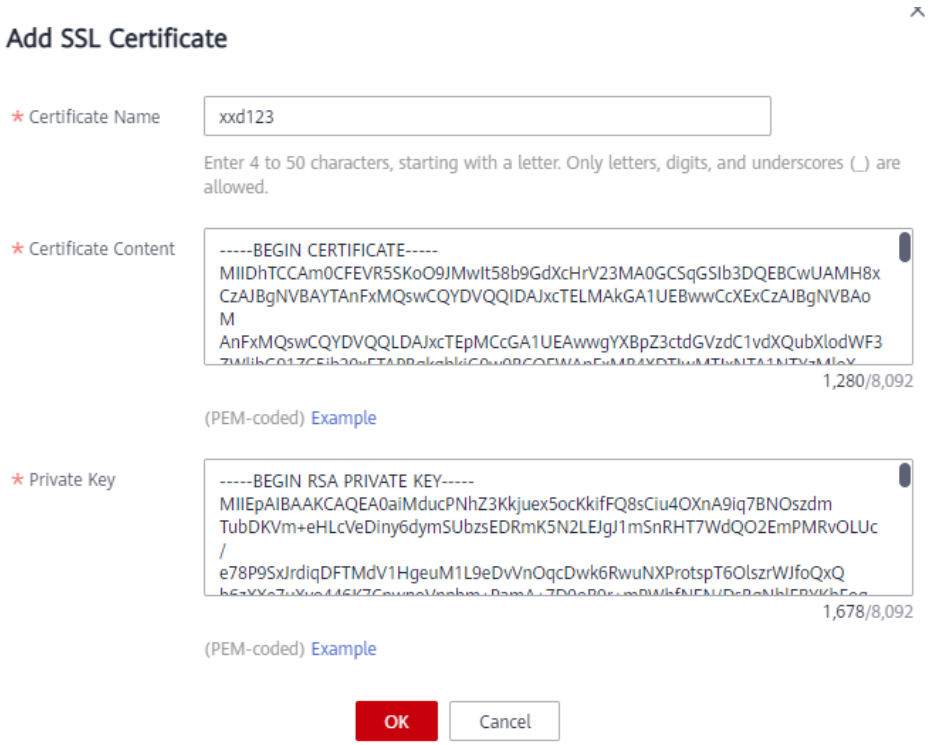
Step 5 Click **OK**.

If the domain name is not needed, click **Unbind** to unbind it from the API group.

Step 6 (Optional) If the API group contains APIs that are accessed through HTTPS, add an SSL certificate.

1. Click **Add SSL Certificate**.
2. Enter the name, content, and key of the **obtained SSL certificate**, and click **OK**.

Figure 11-5 Adding an SSL certificate



NOTE

- Currently, you can only add SSL certificates in the PEM format. To add SSL certificates of other formats, convert the certificates into the PEM format first.
- To replace or edit an SSL certificate, click  next to the certificate name. The certificate content and key will not be visible after you click **OK** to add the certificate. If the content has been updated, add the entire content or key again.
- If you do not require an SSL certificate, click **Delete SSL Certificate** in the row containing the certificate to delete it.

----End

Troubleshooting

- Failure in binding an independent domain name: The independent domain name is not CNAMEd to the subdomain name of the API group, or the independent domain name already exists.
- Failure in adding an SSL certificate: The domain name of the SSL certificate is different from the domain name for which you add the SSL certificate.

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**.

11.3.3 Deleting an API Group

Scenario

You can delete an API group if you do not require it.

NOTE

API groups that contain APIs cannot be deleted.

Prerequisites

You have created an API group.

Procedure

Step 1 **Access the shared gateway console**.

Step 2 In the navigation pane, choose **API Publishing** > **API Groups**.

Step 3 Delete an API group. You can use one of the following methods:

- In the **Operation** column of the target API group, choose **More** > **Delete**.
- Click the name of the target API group, and click **Delete Group** in the upper right corner of the displayed API group details page.

Step 4 Enter **DELETE** and click **Yes**.

----End

11.3.4 Adding a Gateway Response

Scenario

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 gateway responses with custom status codes and content, on the **API Groups** page. 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

- The default gateway responses provided by APIG can be edited.
- You can create gateway responses and configure different responses for APIs in the same API group.
- 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 **APIG Context Variables**.

Prerequisites

You have created an API group.

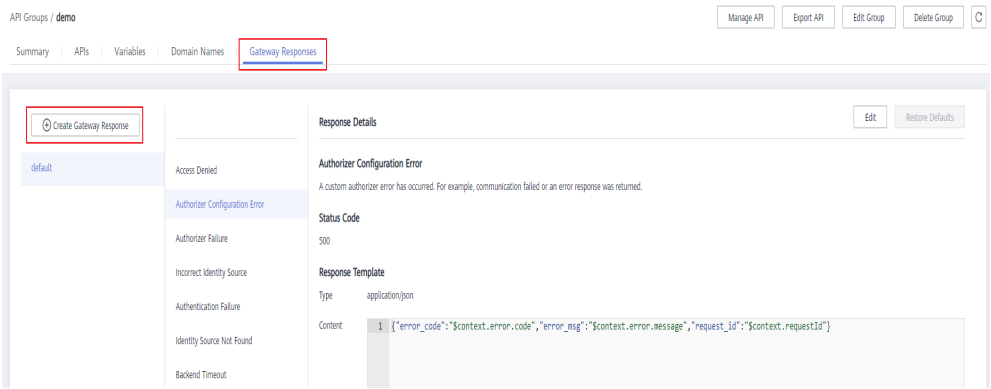
Procedure

Step 1 **Access the shared gateway console**.

Step 2 In the navigation pane, choose **API Publishing** > **API Groups**.

Step 3 Locate the API group for which you want to create or modify a gateway response, and click the group name to go to the API group details page.

Step 4 Click the **Gateway Responses** tab and create a gateway response.



NOTE

- To edit a response, click the **Edit** button in the upper right corner and modify the status code and content of the response.
- You can modify only the status code and content of a default or custom gateway response, and you cannot change the response type.
- Error information and other response details can be obtained using variables. For details about the supported variables, see **Table 11-3**.

----End

Response Types

Table 11-2 lists the response types supported by APIG. You can define status codes of responses to meet your service requirements.

Table 11-2 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.
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.
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.

Response Name	Default Status Code	Description
Unauthorized App	401	The app you are using has not been authorized to call the API.

APIG Context Variables

Table 11-3 Variables that can be used in response message body

Variable	Description
$context.apiId	API ID.
$context.appId	ID of the app 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.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.
$context.error.code	Error code.
$context.error.type	Error type.

11.4 API Management

11.4.1 Creating an API

Scenario

You can selectively expose your services by configuring their APIs in APIG.

To create an API, set the basic information and define the API request, backend service, and responses.

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 during API creation.
- If the backend service of the API is deployed in a VPC, you have created a VPC channel to access the service by following the procedure in **Creating a VPC Channel**. You can also create a VPC channel during API creation.

Setting Basic Information

Step 1 **Access the shared gateway console**.

Step 2 In the navigation pane, choose **API Publishing** > **APIs**.

Step 3 Click **Create API**, and set the parameters listed in **Table 11-4**.

Table 11-4 Basic information

Parameter	Description
Name	API name. It is recommended that you enter a name based on naming rules to facilitate search.
API Group	The group to which the API belongs. If no API group is available, click **Create API Group** to create one.
Gateway Response	Displayed if APIG fails to process an API request. APIG provides a set of default responses and also allows you to **create gateway responses** with custom status codes and content, on the **API Groups** page. The response content must be in JSON format.
Visibility	Determine whether the API is available to the public. Options: • **Public**

Parameter	Description
Security Authentication	The following authentication modes are available: • **App**: Requests for the API will be authenticated by APIG. • **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**. App authentication is recommended. **NOTICE** • If you set the authentication mode of an API to **IAM**, 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 of an API to **None**, any user can access the API over public networks, which can result in excessive charges if the API is bombarded with malicious requests. • If you set the authentication mode of an API to **Custom**, you can create a function in FunctionGraph to interconnect with your own authentication system or service. This authentication mode is not supported in regions where FunctionGraph is unavailable.
Simple Authentication	This parameter is available only if you set **Security Authentication** to **App**. If you select app authentication, you can 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 only supports HTTPS 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**.
Custom Authorizer	This parameter is mandatory if **Security Authentication** is set to **Custom**. Select a custom authorizer if you set **Security Authentication** to **Custom**. If no custom authorizer is available, click **Create Custom Authorizer** to create one.
Tag Name	Classification attribute used to quickly identify the API from other APIs.
Description	Description of the API.

Step 4 Click **Next**.

----End

Defining API Request

Step 1 On the **Define API Request** page, set the parameters listed in **Table 11-5**.

Figure 11-6 Define API Request

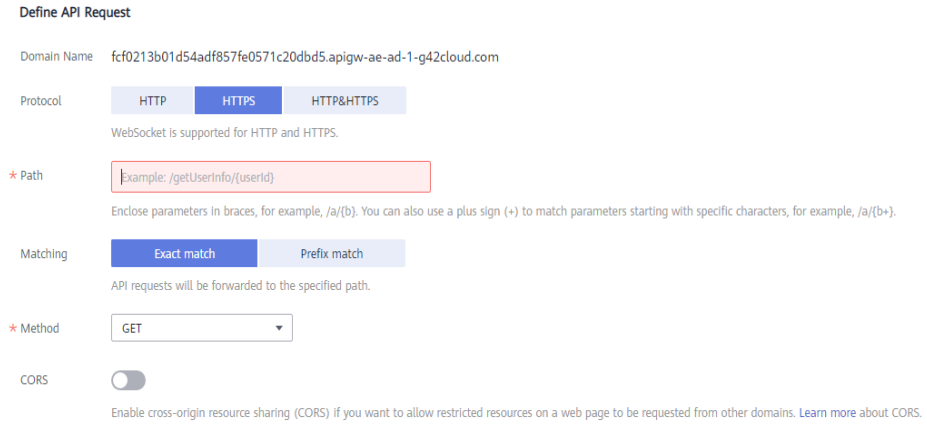


Table 11-5 Parameters for defining API requests

Parameter	Description
Domain Name	The subdomain automatically allocated to the API group.
Protocol	The protocol used for calling the API. Options: • HTTP • HTTPS • 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).

Parameter	Description
Path	The path for requesting the API. Enter a path in the format of "/users/*{userId}*/projects". • The variable in braces ({}) is a request parameter. Ensure that it is an entire segment between a pair of slashes (/). A segment that is not marked by a pair of slashes, for example, **/abc{userId}**, is not supported. If you set the matching mode to **Exact match**, you can add a plus sign (+) to the end of the request parameter, for example, **/users/{*p*+}**. The variable *p* matches the segments between one or multiple pairs of slashes (/). • Ensure that you define the parameters contained in the request path as input parameters. • The content is case-sensitive.
Matching	Options: • **Exact match**: The API can be called only using the specified request path. • **Prefix match**: The API can be called using paths starting with the matching characters. For example, if you set the request path to **/test/AA** and the matching mode to **Prefix match**, the API can be called using **/test/AA/CC** but cannot be called using **/test/AACC**. **NOTE** • Exact match takes precedence over prefix match. Prefix match with a short prefix has a lower priority. For example, for request path **/a/b/c** (exact match), **/a** (prefix match), and **/a/b** (prefix match), the matching order is **/a/b/c** > **/a/b** > **/a**. • 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 service. 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 service. The request URL received by the backend service is **/test2/AA/CC/**.
Method	The API calling method. The options are **GET**, **POST**, **DELETE**, **PUT**, **PATCH**, **HEAD**, **OPTIONS**, and **ANY**. • **ANY** indicates that the API can be called using any request method. • If you set **Method** to **POST**, **PUT**, **PATCH**, or **ANY**, set the request body.

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 you enable CORS, you need to create another API that uses the OPTIONS method. For details, see **CORS**.
Body	Available for **Method** set to **POST**, **PUT**, **PATCH**, or **ANY**. Enter the description of the request body in the API request to help API callers understand how to correctly encapsulate API requests.

Step 2 (Optional) Set input parameters.

Input parameters are transmitted together with the request when the API is called.

1. Click **Add Input Parameter**.
2. Set the parameters listed in **Table 11-6**.

Table 11-6 Input parameter definition

Parameter	Description
Name	Name of the input parameter. If you set the parameter location to **PATH**, ensure that the parameter name is the same as that defined in the request path. **NOTE** – The parameter name is not case-sensitive. 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 (_).
Location	Position of the parameter in requests. The options are **PATH**, **HEADER**, and **QUERY**. **NOTE** If you set the parameter location to **PATH**, you must include the parameter in the request path.

Parameter	Description
Type	Type of the parameter value. Options: **STRING** and **NUMBER**. **NOTE** Set the type of Boolean parameters to **STRING**.
Mandatory	Determine whether the input parameter is required in each request sent to call the API. If you select **Yes**, API requests that do not contain the input parameter will be rejected.
Passthrough	Determine whether to transparently transmit the input parameter to the backend service.
Default Value	The value that will be used if no value is specified for the input parameter when the API is called. If the input parameter is not specified in a request, APIG will automatically send the default value to the backend service.
Enumerated Value	Enumerated value of the input parameter. Use commas (,) to separate multiple enumerated values. The value of this input parameter can only be one of the enumerated values.
Minimum Length	The minimum length of the parameter value. Only numbers are allowed.
Maximum Length	The maximum length of the parameter value. Only numbers are allowed.
Example	Example value for the parameter.
Description	Description of the parameter.

3. Click **OK**.

Step 3 Click **Next**.

----End

Defining Backend Service

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 **Define Backend Request** page, select a backend type.

Table 11-7, Table 11-8, and Table 11-9 describe the backend service parameters.

Table 11-7 Parameters for defining an HTTP/HTTPS backend service

Parameter	Description
Protocol	HTTP or HTTPS. This protocol must be the one used by the backend service. **NOTE** • APIG supports WebSocket data transmission. HTTP is equivalent to WebSocket (ws), and HTTPS is equivalent to WebSocket Secure (wss). • HTTPS is recommended for transmitting important or sensitive data.
Method	The API calling method. The options are **GET**, **POST**, **DELETE**, **PUT**, **PATCH**, **HEAD**, **OPTIONS**, and **ANY**. **ANY** indicates that the API can be called using any request method.
VPC Channel	Determine whether the backend service will be accessed using a VPC channel. • **If yes, select a VPC channel.** To ensure a successful health check and service availability, configure the security groups of cloud servers in each VPC channel to allow access from 100.125.0.0/16. • **If no, configure the backend service address.** Enter a backend address in the format of "backend service 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. 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#**.
Host Header (if applicable)	This parameter is available only if you set **VPC Channel** to **Configure**. Define a host header for requests to be sent to cloud servers associated with the VPC channel. By default, the original host header in each request will be used.
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#**.
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.

Parameter	Description
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 11-8 Parameters for defining a FunctionGraph backend service

Parameter	Description
FunctionURN	Identifier of the requested function. Click **Select Function URN** to specify a function URN.
Version/Alias	Select a function version or alias. For details, see sections "Managing Versions" and "Managing Aliases" in the *FunctionGraph User Guide*.
Invocation Mode	• **Synchronous**: synchronous invocation. 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**: asynchronous invocation. 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)	Backend request timeout. For details, see **Table 11-7**.
Backend Authentication	For details, see the description about backend authentication in **Table 11-7**.

Table 11-9 Parameters for defining a Mock backend service

Parameter	Description
Status Code	HTTP status code for API response. If your gateway does not support status codes, contact technical support to upgrade it.

Parameter	Description
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 11-7**.
Header Parameters	API response headers. Click **Add Header**, and enter the parameter name, value, and description.

NOTE

- If you have defined an environment variable in the backend request path, the API cannot be debugged on the API debugging page.
- For variables defined in the backend request path of an API, corresponding environment variables and their values must be configured. Otherwise, the API cannot be published because there will be no values that can be assigned to the variables.
- Environment variable names are case-sensitive.

Step 2 (Optional) Add a backend policy.

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

1. Click **Add Backend Policy**.
2. Set parameters by referring to **Table 11-10** and **Table 11-7**.

Figure 11-7 Adding a backend policy

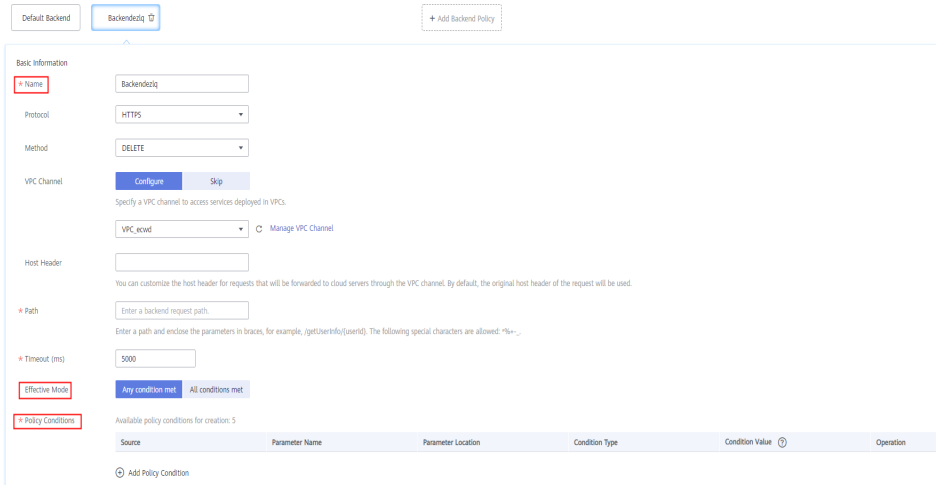


Table 11-10 Backend policy parameters

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 11-11**.

Table 11-11 Policy conditions

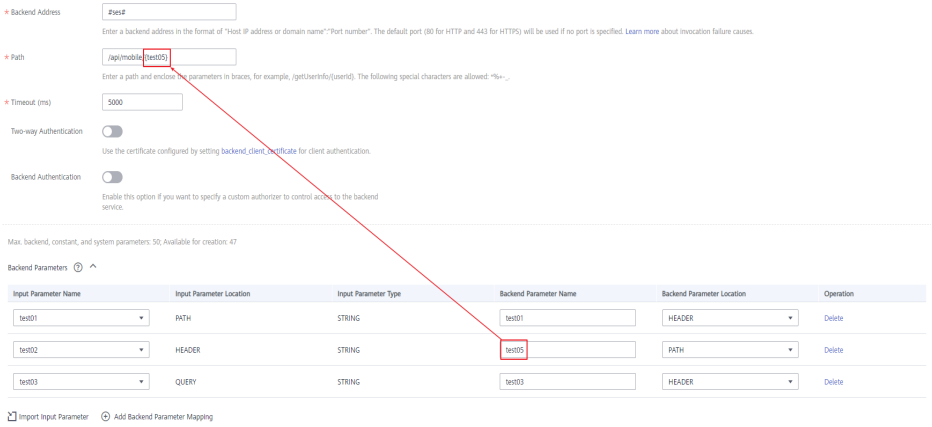
Parameter	Description
Source	– Source IP address – Input parameter **NOTICE** Input parameters (for example, headers) set as policy conditions must have already been defined in the API request settings.
Parameter Name	– When setting **Source** to **Input parameter**, select an input parameter.
Parameter Location	The parameter location is displayed only if you set **Source** to **Input parameter**.
Condition Type	This parameter is required only if you set **Source** to **Input parameter**. – **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.
Condition Value	Set a condition value according to the condition type. – **Equal**: Enter a value. – **Enumerated**: Enter multiple values and separate them using commas. – **Matching**: Enter a range, for example, **[0-5]**. If you have set **Source** to **Source IP address**, enter one or more IP addresses and separate them using commas.

Step 3 (Optional) Set backend parameters.

Input parameters of the API are mapped to corresponding backend parameters in backend requests.

1. Click next to **Backend Parameters**, and define backend parameters. You can use one of the following methods:
   - Click **Import Input Parameter**. All the defined input parameters are automatically displayed.
   - Click **Add Backend Parameter Mapping**, and add required backend parameters.
2. Modify the mappings based on the parameters and their locations in backend requests. **Figure 11-8** highlights the backend parameters.

   Figure 11-8 Backend service parameters

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

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

      NOTE

      - The parameter name is not case-sensitive. 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 contain underscores (_).

   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.

      For example, **test01** is **abc**, **test02** is **def**, and **test03** is **xyz**.

      API request:

      ```
      curl -ik -H 'test02:def' -X GET https://www.example01.com/v1.0/abc?test03=xyz
      ```

      Backend request:

      ```
      curl -ik -H 'test01:abc' -H 'test03:xyz' -X GET https://www.example02.com/v1.0/def
      ```

Step 4 (Optional) Set constant parameters.

You can define constant parameters for the backend service to receive constants that are invisible to API callers. APIG adds constant parameters to specified positions in the request sent to the backend service.

NOTICE

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

1. Click ⌄ next to **Constant Parameters**.
2. Click **Add Constant Parameter**, and set the parameters listed in **Table 11-12**.

Figure 11-9 Adding constant parameters

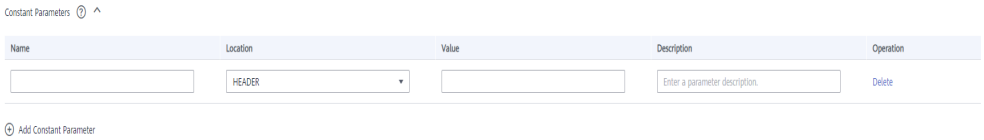


Table 11-12 Setting constant parameters

Parameter	Description
Name	Constant parameter name. If you set the parameter location to **PATH**, ensure that the parameter name is the same as that defined in the backend request path. **NOTE** – The parameter name is not case-sensitive. 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 contain underscores (_).
Location	Position of the parameter in requests. The options are **PATH**, **QUERY**, and **HEADER**.
Value	Value of the parameter.
Description	Description of the constant parameter.

NOTE

- APIG sends requests containing constant parameters to backend services after percent-encoding of special parameter values. Ensure that the backend services support percent-encoding. For example, parameter value **[apig]** becomes **%5Bapig%5D** after percent-encoding.
- For values of path parameters, the following characters will be percent-encoded: ASCII codes 0–31, blank symbols, ASCII codes 127–255, and special characters ?></%#"[\]^`{|}
- For values of query strings, the following characters will be percent-encoded: ASCII codes 0–31, blank symbols, ASCII codes 127–255, and special characters >=<+&%#"[\]^`{|}

Step 5 (Optional) Set system parameters.

System parameters refer to runtime parameters regarding gateway running and frontend and backend authentications. The parameters are transferred to the API backend service for access control and custom authentication.

1. Click ⌄ next to **System Parameters**.
2. Click **Add System Parameter**, and set the parameters listed in **Table 11-13**.

Figure 11-10 Adding a system parameter

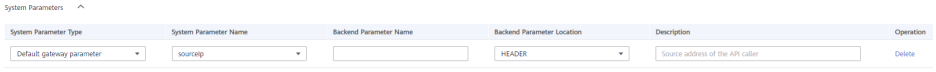


Table 11-13 System parameters

Parameter	Description
System Parameter Type	– **Default gateway parameter**: Default parameters supported by APIG. – **Frontend authentication parameter**: Parameters to be displayed in the frontend custom authentication result. This option is available only if you select **Custom** for **Security Authentication** on the **Set Basic Information** page. – **Backend authentication parameter**: Parameters to be displayed in the backend custom authentication result. This option is available only if you enable **Backend Authentication** on the **Define Backend Request** page.

Parameter	Description
System Parameter Name	– If **System Parameter Type** is **Default gateway parameter**, select any of the following parameters. ▪ **sourceIp**: source IP address of the 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**: app ID of the API provider – Ensure that the frontend and backend authentication parameters are consistent with the return result parameters defined for the corresponding custom authorizer function. For details about how to create a custom authorizer function and obtain returned result parameters, see **API Gateway Developer Guide**.
Backend Parameter Name	Name of the backend parameter to which the system parameter will be mapped. **NOTE** – The parameter name is not case-sensitive. 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 contain underscores (_).
Backend Parameter Location	Position of the backend parameter in requests.
Description	Description of the system parameter.

Step 6 Click **Next**.

----End

Defining Responses

Step 1 On the **Define Response** page, set the parameters listed in **Table 11-14**.

Table 11-14 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 2 Click **Finish**.

After the API is created, click its name in the API list to view details.

----End

FAQs About API Creation

Does APIG Support Multiple Backend Endpoints?

What Are the Possible Causes If a Backend Service Fails to Be Invoked or the Invocation Times Out?

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

Follow-Up Operations

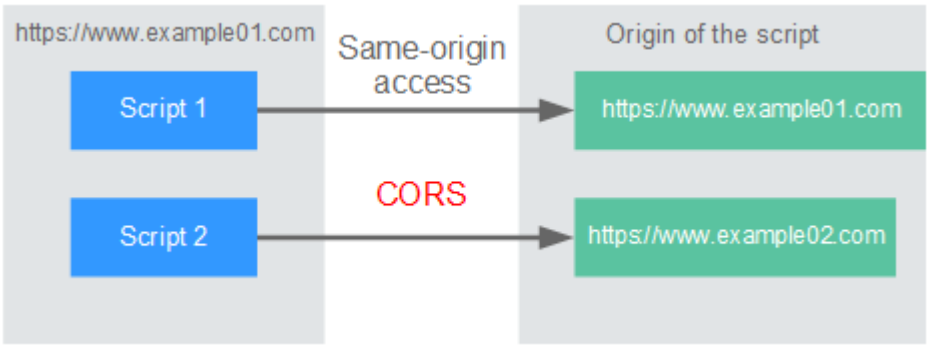
After creating an API, verify it by following the procedure in **Debugging an API**.

11.4.2 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 11-11 CORS



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.

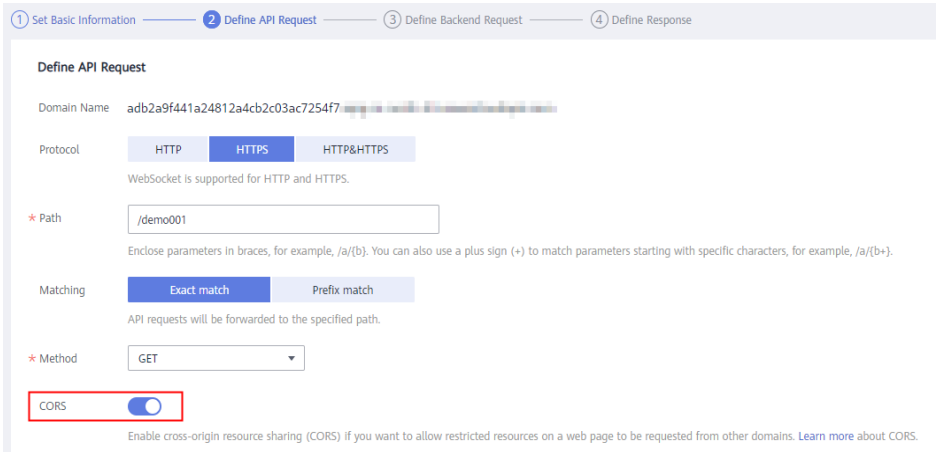
Configuring CORS

CORS is disabled by default. To enable CORS for an API, perform the operations described in this section.

- **Simple CORS requests**

  When creating an API, enable CORS on the API request configuration page. For more information, see **Simple Request**.

Figure 11-12 CORS



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

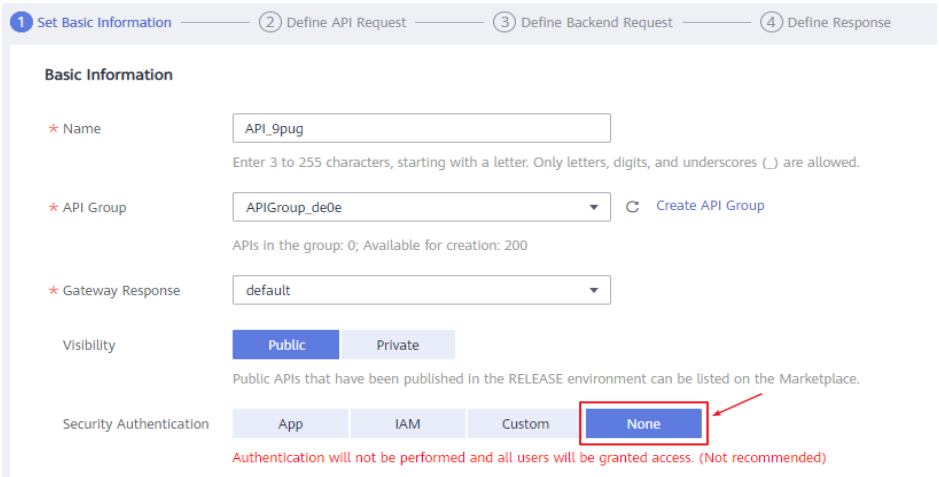
NOTICE

If your API will receive not-so-simple requests, **create another API that will be accessed using the OPTIONS method** in the same group as the target API to receive preflight requests.

Follow this procedure to define the preflight request API. For more information, see **Not-So-Simple Request**.

a. On the **Set Basic Information** page, select **None** to skip over security authentication.

Figure 11-13 Preflight request - None authentication

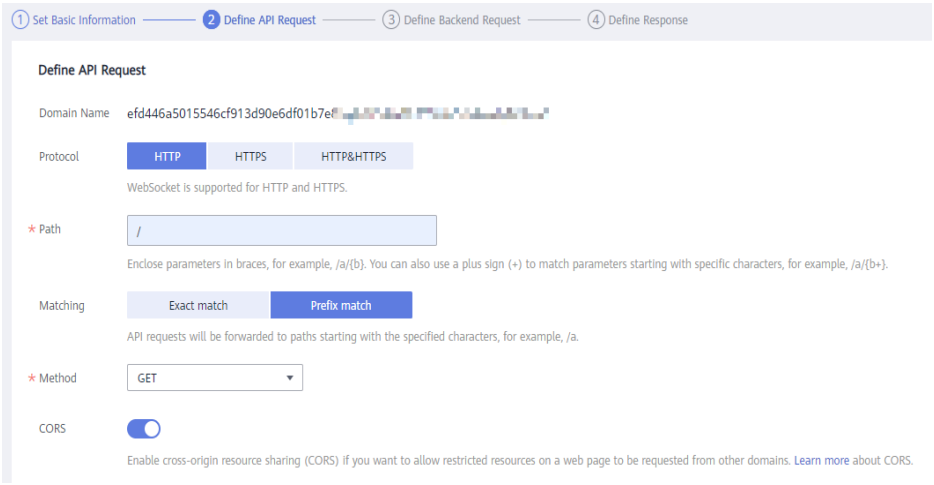


b. On the **Define API Request** page, perform the following settings:

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

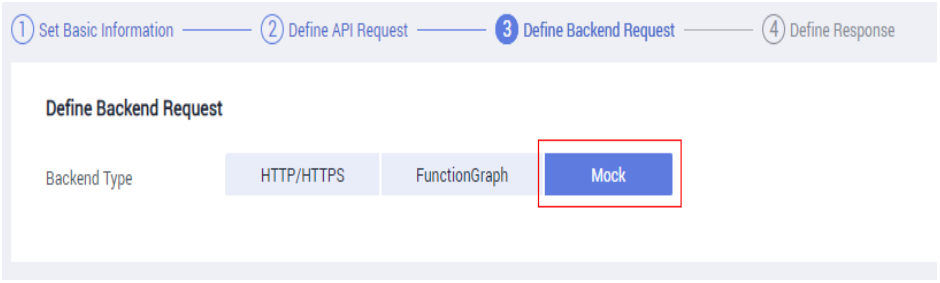
- **CORS**: Enabled.

Figure 11-14 Preflight request - Defining API requests



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

Figure 11-15 Preflight request - Mock backend service



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.

NOTE

If you use the CORS plug-in for an API, you do not need to create another API that uses the OPTIONS method.

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

- **API Group**: The same group to which the API with CORS enabled belongs.
- **Security Authentication**: Select **None**. No authentication is required for requests received by the new API no matter which security authentication mode has been selected.
- **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.
- **Method**: Select **OPTIONS**.
- **CORS**: Enabled.

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"}
```

11.4.3 Debugging an API

Scenario

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

NOTE

- APIs with backend request paths containing variables cannot be debugged.
- If an API has been bound with a request throttling policy, the policy will not work during debugging of the API.

Prerequisites

- You have created an API group and API.
- You have set up the backend service of the API.

Procedure

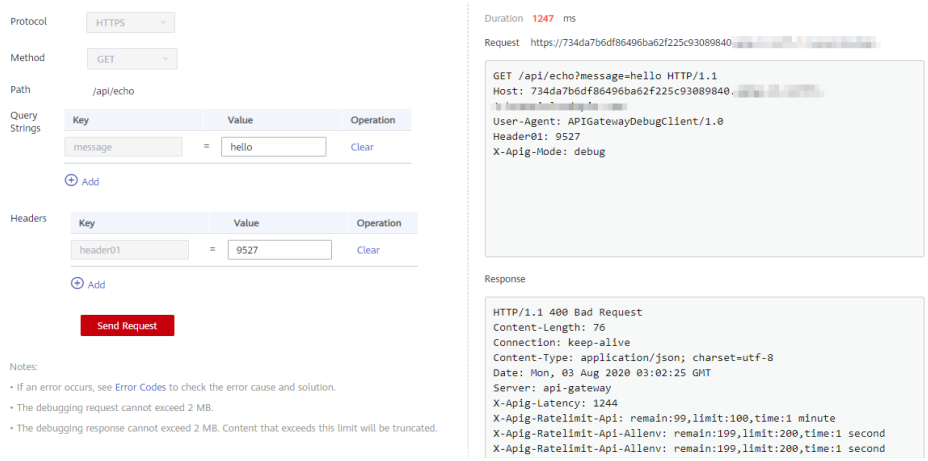
Step 1 Access the shared gateway console.

Step 2 In the navigation pane, choose **API Publishing** > **APIs**.

Step 3 Debug an API. You can use one of the following methods:

- In the **Operation** column of the API you want to debug, choose **More** > **Debug**.
- Click the name of the target API, and click **Debug** in the upper right corner of the displayed API details page.

Figure 11-16 Debugging an API



On the left side, set the API request parameters listed in Table 11-15. On the right side, view the API request and response information after you click **Send Request**.

Table 11-15 Parameters for debugging an API

Parameter	Description
Protocol	This parameter can be modified only if you set **Protocol** to **HTTP&HTTPS** for the API.
Method	This parameter can be modified only if you set **Method** to **ANY** for the API.
Suffix	You can define a path only if you have set **Matching** to **Prefix match** for the API.
Path	Request path of the API.
Path Parameters	This parameter can be modified only if you have defined path parameters (such as **{test}**) for the API.

Parameter	Description
Query Strings	Query string parameters and values.
Headers	HTTP headers and values.
Body	This parameter can be modified only if you set **Method** to **PATCH**, **POST**, or **PUT** for the API.

NOTE

The fields displayed on the debugging page vary according to the request type.

Step 4 After setting request parameters, click **Send Request**.

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

- If the debugging is successful, the HTTP status code **200** 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 5 You can send more requests with different parameters and values to verify the API.

NOTE

To modify the API configurations, click **Edit** in the upper right corner, and modify the parameters on the **Edit API** page.

----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 of the API, create request throttling policies (see **Creating a Request Throttling Policy**), access control policies (**Creating an Access Control Policy**), and signature keys (**Creating and Using a Signature Key**) for the API.

11.4.4 Authorizing Apps to Call an API

Scenario

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

NOTE

- You can only authorize apps to call published APIs.
- You can authorize apps only to call APIs that use app authentication.

Prerequisites

- You have created an API group and API.

- (Optional) You have created an environment.
- You have created an app.

Procedure

Step 1 Access the shared gateway console.

Step 2 In the navigation pane, choose **API Publishing** > **APIs**.

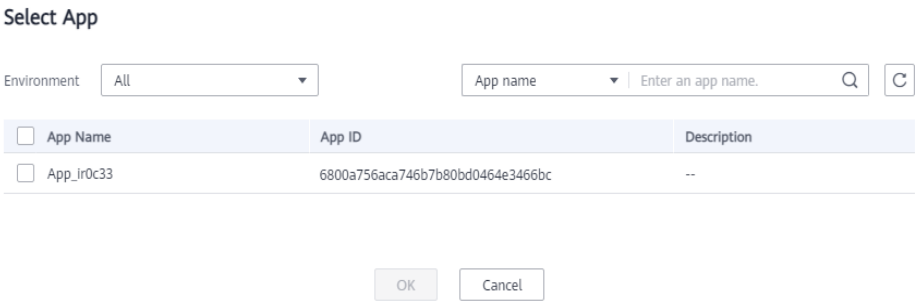
Step 3 Authorize apps to call an API. You can use one of the following methods:

- In the **Operation** column of the target API, choose **More** > **Authorize App**, and then click **Select App**.
- Select the target API, click **Authorize App** over the API list, and then click **Select App**.
- Authorize apps through the API details page.
  a. Click the name of the target API.
  b. Click the **Authorization** tab.
  c. Click **Select App**.

NOTE

To authorize an app to access multiple APIs, select the APIs, and click **Authorize App**. Click **Select App**, select the app you wish to authorize, and click **OK**. You can grant access to a maximum of 1000 APIs at a time.

Step 4 Select an environment, search for and select desired apps, and click **OK**.



Step 5 After the authorization is complete, view the authorized apps on the **Authorization** tab page or the **Authorize App** page.

NOTE

If an app does not need to call the API, click **Cancel Authorization** in the row containing the app to unbind it.

----End

Follow-Up Operations

After you authorize an app to call an API, the API can be called using SDKs of different programming languages.

11.4.5 Publishing an API

Scenario

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 API group and API.
- You have created an environment.

Publishing an API

Step 1 Access the shared gateway console.

Step 2 In the navigation pane, choose **API Publishing** > **APIs**.

Step 3 Publish an API. You can use one of the following methods:

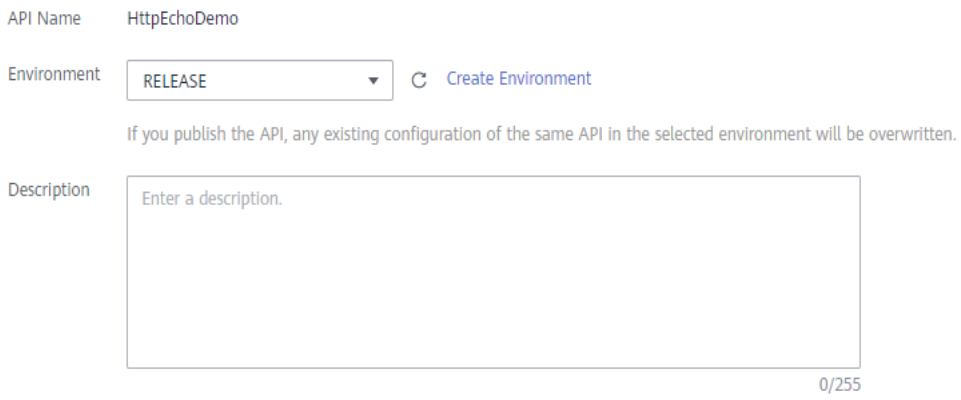
- Click **Publish** in the row containing the API you want to publish.
- Click the name of the target API, and click **Publish** in the upper right corner of the displayed API details page.

NOTE

To publish multiple APIs, select the APIs, and click **Publish**. You can publish a maximum of 1000 APIs at a time.

Step 4 Select the environment where the API will be published, and enter a description.

Figure 11-17 Publishing an API



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 5 Click **Publish**.

----End

Viewing Publication History

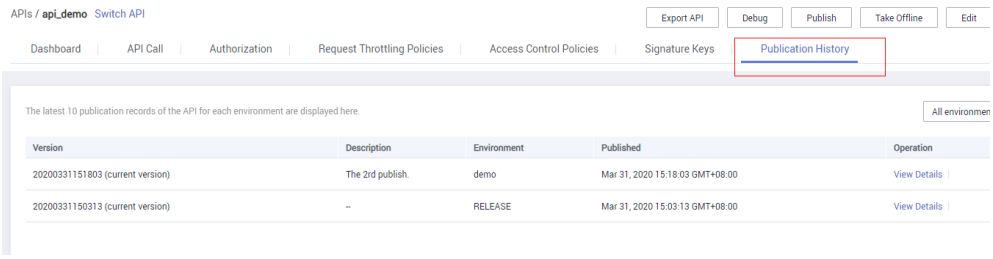
Step 1 In the navigation pane, choose **API Publishing** > **APIs**.

Step 2 Click the name of the target API.

Step 3 Click the **Publication History** tab.

The publication history of the API is displayed.

Figure 11-18 Viewing publication history



Step 4 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 5 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?

Why Can't APIs Published in a Non-RELEASE Environment Be Accessed?

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

11.4.6 Taking an API Offline

Scenario

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 **Access the shared gateway console**.

Step 2 In the navigation pane, choose **API Publishing** > **APIs**.

Step 3 Take the API offline. You can use one of the following methods:

- In the **Operation** column of the target API, choose **More** > **Take Offline**.
- Click the name of the target API, and click **Take Offline** in the upper right corner of the API details page.

NOTE

To take multiple APIs offline, select the APIs, and click **Take Offline**. You can take a maximum of 1000 APIs offline at a time.

Step 4 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 based on the instructions provided in **Deleting an API**.

11.4.7 Deleting an API

Scenario

You can delete published APIs you no longer require.

NOTICE

- Deleted APIs cannot be accessed by apps or users who were using the APIs, so make sure you notify users before the deletion.
- Published APIs must be first taken offline and then deleted.

Procedure

Step 1 Access the shared gateway console.

Step 2 In the navigation pane, choose **API Publishing** > **APIs**.

Step 3 Delete the API. You can use one of the following methods:

- In the **Operation** column of the API you want to delete, choose **More** > **Delete**.
- Click the name of the target API, and click **Delete** in the upper right corner of the displayed API details page.

NOTE

To delete multiple APIs, select the APIs, and click **Delete**. You can delete a maximum of 1000 APIs at a time.

Step 4 Enter **DELETE** and click **Yes**.

----End

11.4.8 Importing APIs

Scenario

APIG allows you to import Swagger 2.0 APIs to existing or new API groups. Swagger is an open-source tool built based on OpenAPI specifications to design, build, record, and use REST APIs.

You can import APIs individually or in batches depending on the number of APIs contained in a Swagger file.

Prerequisites

- Supplement the **extended Swagger definition** of the APIs to import. If the extended definition does not contain the required settings, create them on the APIG console.
- You have sufficient API group and API quotas.

Procedure

Step 1 Access the shared gateway console.

Step 2 In the navigation pane, choose **API Publishing** > **APIs**.

Step 3 Click **Import API**.

Step 4 Set the parameters listed in **Table 11-16**.

Figure 11-19 Importing APIs

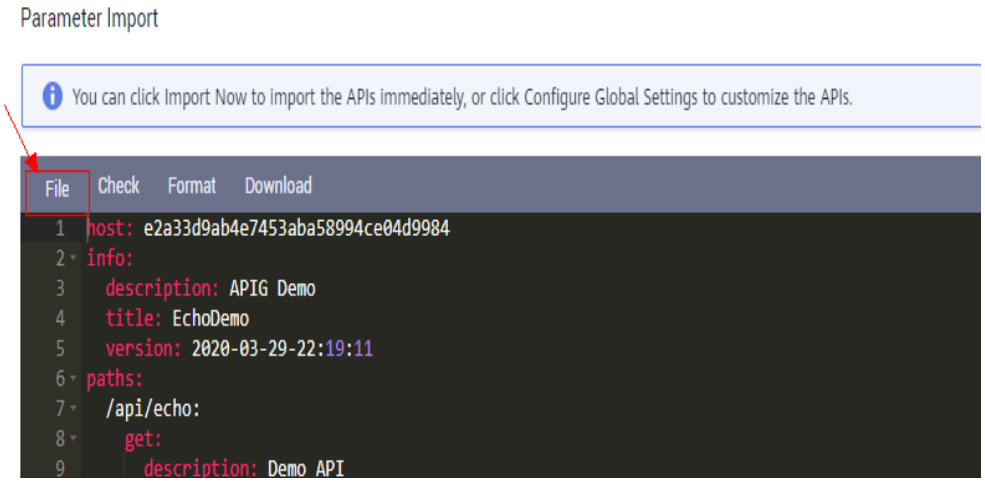
Table 11-16 Parameters for importing APIs

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 5 In the **Parameter Import** area, click **File** and select a file to import.

YAML and JSON files are supported. You can preview the API content to be imported on the **Import API** page.

Figure 11-20 Parameter Import



Step 6 (Optional) Configure global settings for the APIs to be imported.

You can configure the global settings for the APIs, such as frontend and backend requests, or modify other parameters of the APIs.

Figure 11-21 Configuring global settings

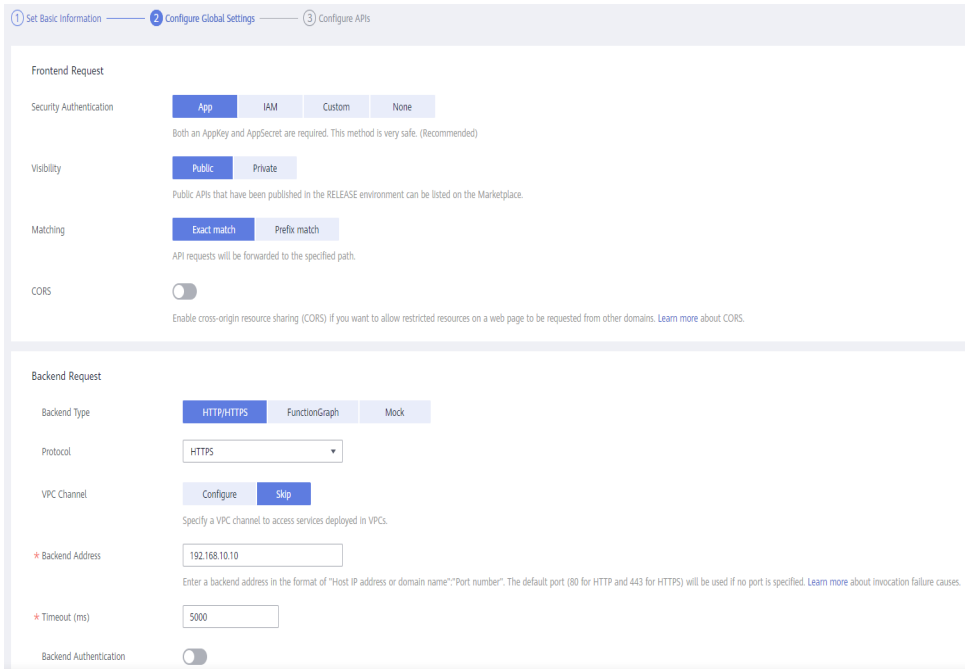
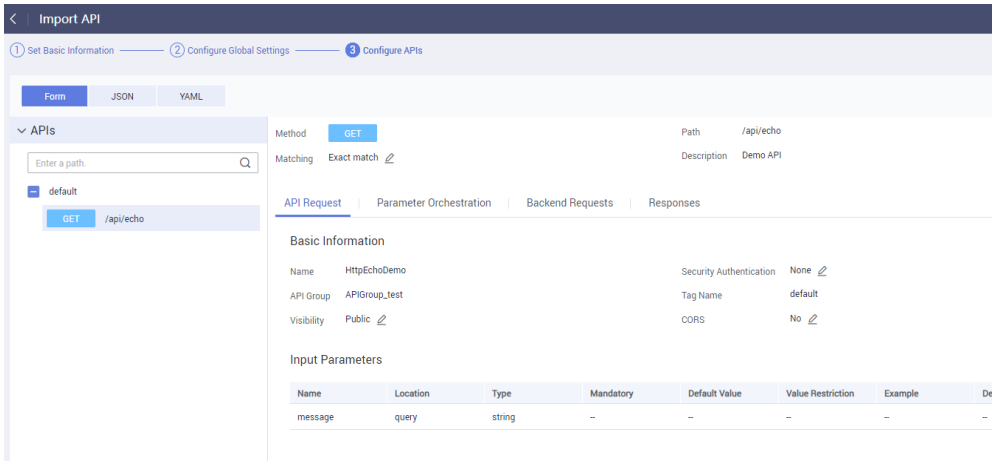


Figure 11-22 Modifying APIs



Step 7 Click **Import Now** to import the APIs.

NOTE

Imported APIs must be manually published so that they become available for users to access.

----End

Follow-Up Operations

Publish the imported API in an environment so that it can be called by users.

11.4.9 Exporting APIs

Scenario

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

Prerequisites

You have created an API group and API.

Procedure

Step 1 **Access the shared gateway console**.

Step 2 Click **Export API**.

Step 3 Set the parameters listed in **Table 11-17**.

Figure 11-23 Exporting APIs

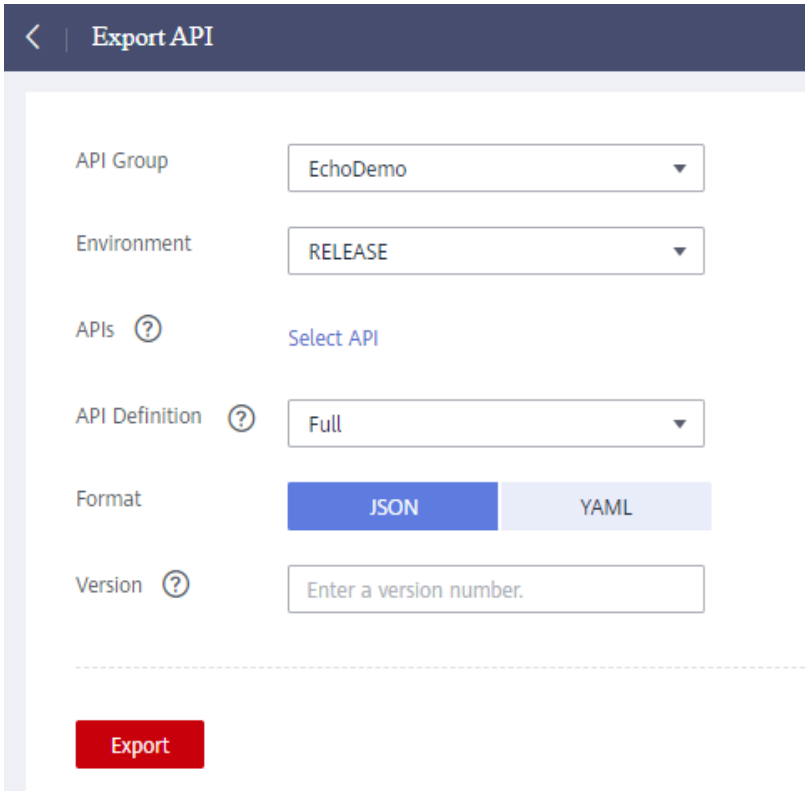


Table 11-17 Parameters for exporting APIs

Parameter	Description
API Group	Select the API group from 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 API group that have been published in the selected environment are exported. To export only specific APIs, click **Select API**, 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. • **Full**: The full definition of an API is composed of the request, backend, and response definitions. • **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	Export APIs in **JSON** or **YAML** format.

Parameter	Description
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.

Step 4 Click **Export**.

The export result is displayed on the right.

----End

11.5 Request Throttling

11.5.1 Creating a Request Throttling Policy

Scenario

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

To provide stable, uninterrupted services, you can create request throttling policies to control the number of calls made to your APIs.

Request throttling policies take effect for an API only if they have been bound to the API.

NOTE

- 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 the shared gateway, the default request throttling limit is 200 calls per second.

Prerequisites

You have **published the API** to which you want to bind a request throttling policy.

Creating a Request Throttling Policy

Step 1 **Access the shared gateway console**.

Step 2 In the navigation pane, choose **API Publishing** > **Request Throttling**.

Step 3 Click **Create Request Throttling Policy**, and set the parameters listed in **Table 11-18**.

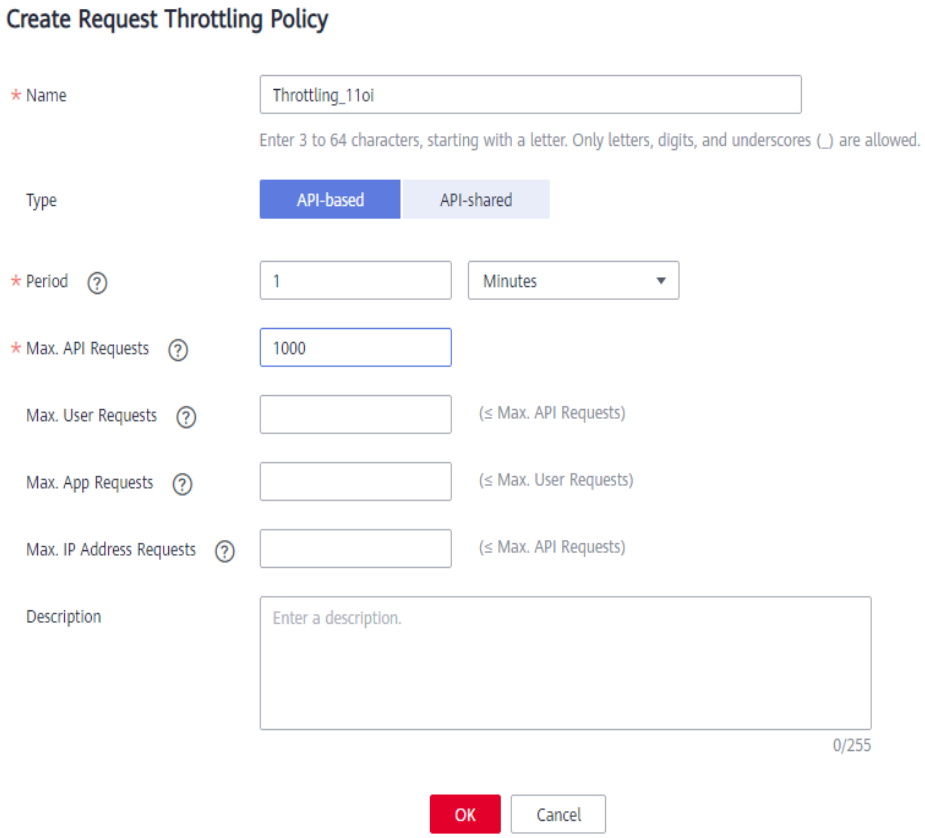


Table 11-18 Parameters for creating a request throttling policy

Parameter	Description
Name	Request throttling policy name.
Type	API-based or API-shared request throttling. • **API-based**: Request throttling is based on every API to which the policy is bound. • **API-shared**: 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. App Requests**: Limit the maximum number of times an API can be called by an app 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.

Parameter	Description
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**.
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. App Requests	The maximum number of times each bound API can be called by an app 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**. • 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. • 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.

Step 4 Click **OK**.

After the policy is created, it is displayed on the **Request Throttling** page. You can bind this policy to APIs to throttle API requests.

----End

Binding a Request Throttling Policy to an API

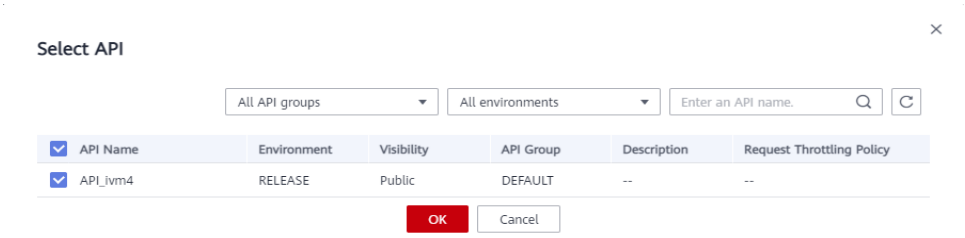
Step 1 Go to the page for binding a request throttling policy to an API. You can use one of the following methods:

- In the **Operation** column of the request throttling policy to be bound, click **Bind to API**, and then click **Select API**.
- Click the name of the target request throttling policy, and click **Select API** on the **APIs** tab page.

Step 2 Specify an API group, environment, and API name keyword to search for the desired API.

Step 3 Select the API and click **OK**.

Figure 11-24 Binding a request throttling policy to an API



NOTE

If a request throttling policy is no longer needed for an API, you can unbind it. To unbind a request throttling policy from multiple APIs, select the APIs, and click **Unbind**. You can unbind a request throttling policy from a maximum of 1000 APIs at a time.

----End

Follow-Up Operations

To control the maximum number of API calls received from a specific app or tenant, specify the app or tenant to exclude by referring to **Adding an Excluded App or Tenant**. If an app is excluded in a request throttling policy, any threshold configured for that app takes precedence over the request throttling policy. The API and user request limits of this policy are still valid. If a tenant is excluded in a request throttling policy, any threshold configured for that tenant will be applied. The API and app request limits of this policy are still valid.

11.5.2 Deleting a Request Throttling Policy

Scenario

You can delete request throttling policies you no longer require.

Prerequisites

You have created a request throttling policy.

Procedure

Step 1 **Access the shared gateway console**.

Step 2 In the navigation pane, choose **API Publishing** > **Request Throttling**.

Step 3 Delete a request throttling policy. You can use one of the following methods:

- In the **Operation** column of the request throttling policy you want to delete, click **Delete**.
- Click the name of the target request throttling policy, and click **Delete** in the upper right corner of the displayed request throttling policy details page.

NOTE

- If a request throttling policy has been bound to an API, unbind the policy and then delete it. To unbind a request throttling policy, go to the policy details page, click **Unbind** in the row that contains the API from which you want to unbind the policy, and click **Yes**.
- To delete multiple request throttling policies, select the policies, and click **Delete**. You can delete a maximum of 1000 request throttling policies at a time.

Step 4 Click **Yes**.

----End

11.5.3 Adding an Excluded App or Tenant

Scenario

If you want to control the number of API calls received from a specific app or tenant, add an excluded app or tenant to a request throttling policy.

Prerequisites

You have created an app or obtained an app ID of another account or an account ID.

Adding an Excluded App

Step 1 Access the shared gateway console.

Step 2 In the navigation pane, choose **API Publishing** > **Request Throttling**.

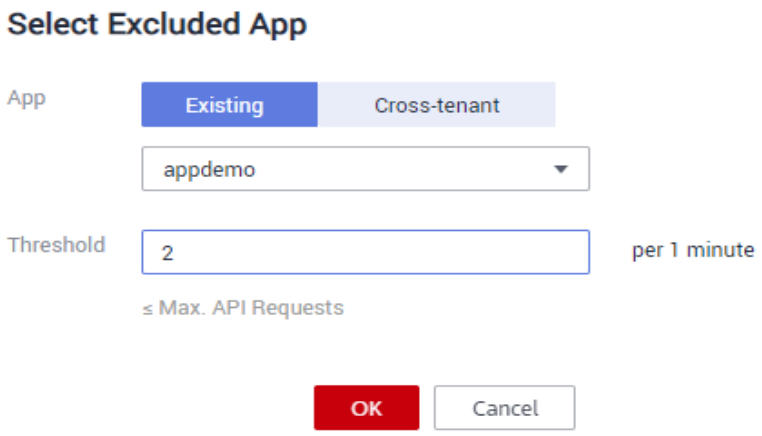
Step 3 Click the name of the target request throttling policy.

Step 4 On the displayed request throttling policy details page, click the **Excluded Apps** tab.

Step 5 Click **Select Excluded App**.

Step 6 Select an app to exclude. You can use one of the following methods:

Figure 11-25 Selecting an app



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

  NOTE

  Excluded app thresholds take precedence over the value of **Max. App Requests**.

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

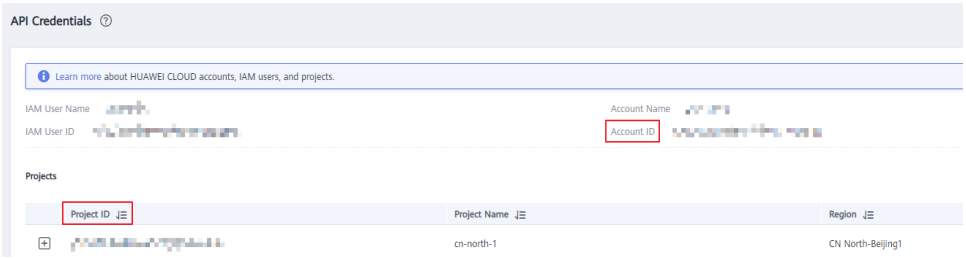
----End

Adding an Excluded Tenant

Step 1 Hover the mouse pointer over the username and choose **My Credentials** from the drop-down list.

Step 2 On the **API Credentials** page, view the account ID and project ID.

Figure 11-26 Viewing the account ID and project ID



Step 3 In the navigation pane, choose **API Publishing** > **Request Throttling**.

Step 4 Click the name of the target request throttling policy.

Step 5 Click the **Excluded Tenants** tab.

Step 6 Click **Select Excluded Tenant**.

Step 7 In the **Select Excluded Tenant** dialog box, set the parameters listed in **Table 11-19**.

Figure 11-27 Adding an excluded tenant

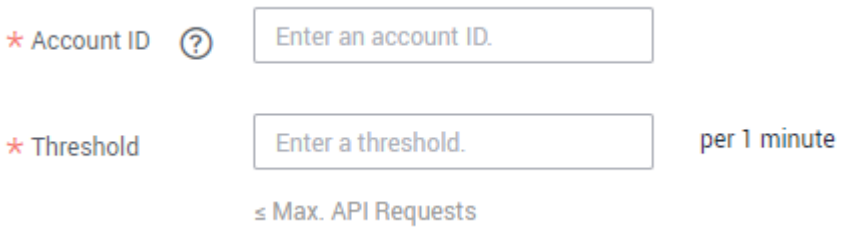


Table 11-19 Excluded tenant configuration

Parameter	Description
Account ID	Account ID or project ID obtained in **Step 2**. • Enter a project ID if you will bind or have bound this policy to an API that uses app authentication. • Enter an account ID if you will bind or have bound this policy to an API that uses IAM authentication.
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 8 Click **OK**.

NOTE

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

For example, suppose a request throttling policy is 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

11.5.4 Removing an Excluded App or Tenant

Scenario

You can remove excluded apps or tenants from a request throttling policy. This section takes an excluded app as an example.

Prerequisites

- You have created a request throttling policy.
- You have already added an excluded app or tenant to the request throttling policy.

Removing an Excluded App

Step 1 Access the shared gateway console.

Step 2 In the navigation pane, choose **API Publishing** > **Request Throttling**.

Step 3 Click the name of the target request throttling policy.

Step 4 Click the **Excluded Apps** tab on the displayed request throttling policy details page.

Step 5 In the **Operation** column of the app you want to remove, click **Remove**.

Step 6 Click **Yes**.

----End

Removing an Excluded Tenant

Step 1 In the navigation pane, choose **API Publishing** > **Request Throttling**.

Step 2 Click the name of the target request throttling policy.

Step 3 Click the **Excluded Tenants** tab.

Step 4 In the **Operation** column of the tenant you want to remove, click **Remove**.

Step 5 Click **Yes**.

----End

11.6 Access Control

11.6.1 Creating an Access Control Policy

Scenario

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 or accounts.

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

NOTE

Each API can be bound with only one access control policy for a given environment, but each access control policy can be bound to multiple APIs.

Creating an Access Control Policy

Step 1 **Access the shared gateway console**.

Step 2 In the navigation pane, choose **API Publishing** > **Access Control**.

Step 3 Click **Create Access Control Policy**.

Step 4 In the **Create Access Control Policy** dialog box, set the parameters listed in **Table 11-20**.

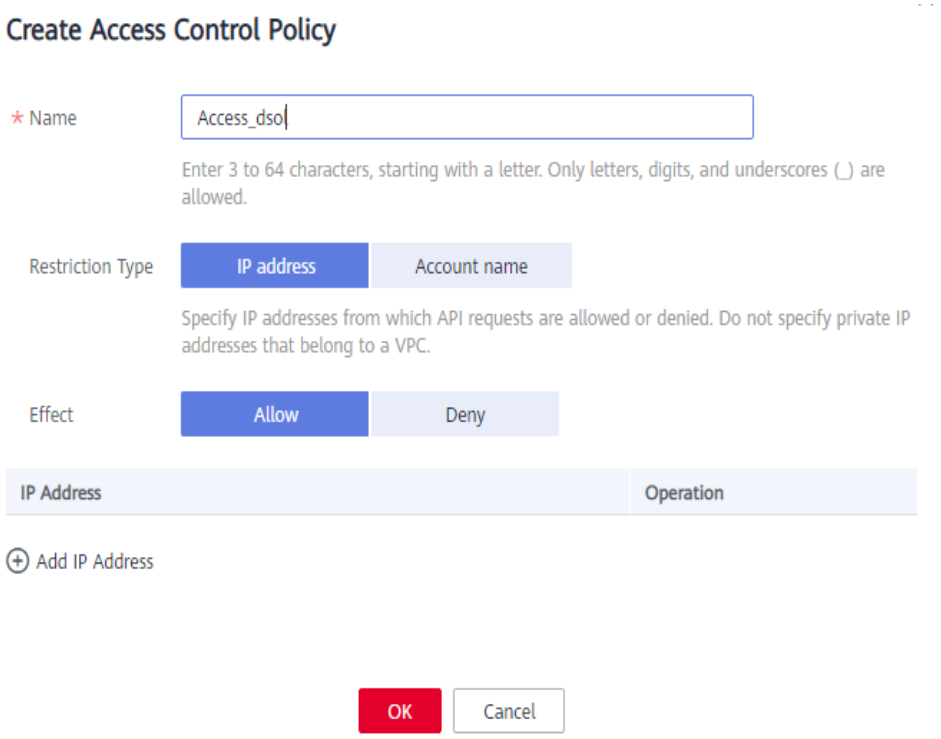


Table 11-20 Parameters for creating an access control policy

Parameter	Description
Name	Access control policy name.
Restriction Type	Type of the source from which API calls are to be controlled. • **IP address**: Specify IP addresses and IP address ranges that are allowed or not allowed to access an API. • **Account name**: Specify names of the accounts that are allowed or not allowed to access an API.
Effect	Options: **Allow** and **Deny**. Use this parameter along with **Restriction Type** to control the access of certain IP addresses or accounts to an API.
IP Address	IP addresses and IP address ranges that are allowed or not allowed to access an API You need to set this parameter only if you have set **Restriction Type** to **IP address**. **NOTE** You can set a maximum of 100 IP addresses respectively to allow or deny access.

Parameter	Description
Account Name	Names of the accounts that are allowed or not allowed to access an API. **This parameter only applies to APIs that are accessed through IAM authentication.** You need to set this parameter only if you have set **Restriction Type** to **Account name**. You can enter multiple account names and separate them with commas, for example, **aaa,bbb**. **NOTE** APIG performs access control on accounts, not IAM users created using accounts.

Step 5 Click **OK**. You can bind the policy to APIs to control API access.

----End

Binding an Access Control Policy to an API

Step 1 Go to the page for binding an access control policy to an API. You can use one of the following methods:

- In the **Operation** column of the access control policy to be bound, click **Bind to API**, and then click **Select API**.
- Click the name of the target access control policy, and click **Select API**.

Step 2 Specify an API group, environment, and API name keyword to search for the desired API.

Step 3 Select the API and click **OK**.

NOTE

If an access control policy is no longer needed for an API, you can unbind it from that API. To unbind an access control policy from multiple APIs, select the APIs, and click **Unbind**. You can unbind a request throttling policy from a maximum of 1000 APIs at a time.

----End

11.6.2 Deleting an Access Control Policy

Scenario

You can delete access control policies you no longer require.

Prerequisites

You have created an access control policy.

Procedure

Step 1 Access the shared gateway console.

Step 2 In the navigation pane, choose **API Publishing** > **Access Control**.

Step 3 Delete an access control policy using one of the following methods:

- In the **Operation** column of the access control policy you want to delete, click **Delete**.
- Click the name of the target access control policy, and click **Delete** in the upper right corner of the displayed access control policy details page.

NOTE

- If an access control policy has been bound to APIs, unbind it and then delete it.
- To delete multiple access control policies, select the policies, and click **Delete**. You can delete a maximum of 1000 access control policies at a time.

Step 4 Click **Yes**.

----End

11.7 Environment Management

11.7.1 Creating an Environment and Environment Variable

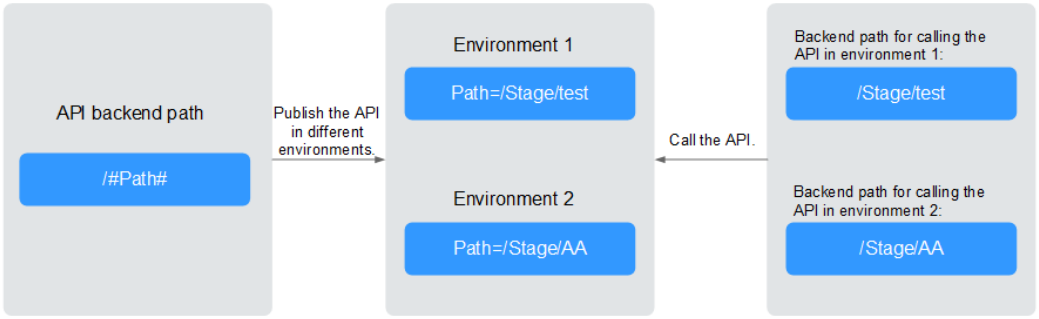
Scenario

An API can be called in different environments, such as production, testing, and development environments. RELEASE is the default environment provided by APIG. 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 create 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 11-28 Environment variables



NOTE

You can create a maximum of 50 variables for an API group in each environment.

Prerequisites

You have **created an API group**.

Creating an Environment

Step 1 **Access the shared gateway console**.

Step 2 In the navigation pane, choose **API Publishing** > **Environments**.

Step 3 Click **Create Environment**, and set the parameters listed in **Table 11-21**.

Figure 11-29 Creating an environment

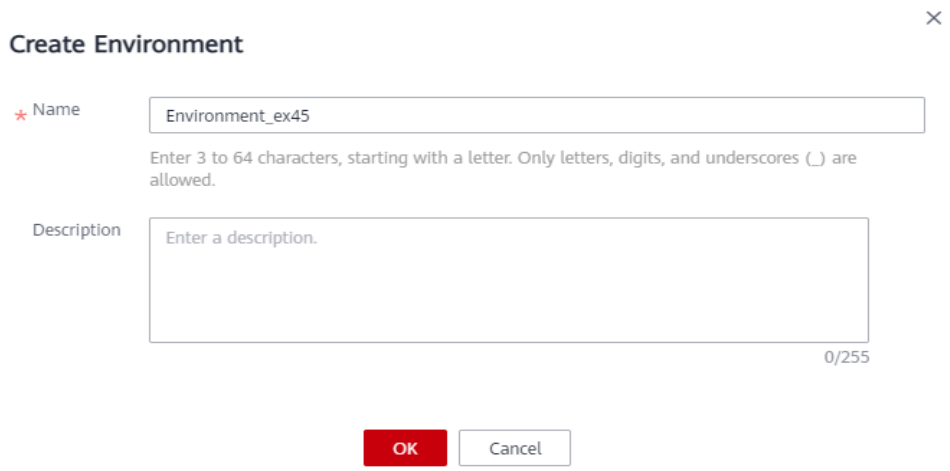


Table 11-21 Environment information

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

Step 4 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 using environment variables.

Creating an Environment Variable

Step 1 In the navigation pane, choose **API Publishing** > **API Groups**.

Step 2 Create a variable. You can use one of the following methods:

- Click the name of the target API group, and click the **Variables** tab on the displayed API group details page.
- In the **Operation** column of the target API group, choose **More** > **Manage Variable**.

Step 3 Select an environment from the **Environment** drop-down list, and click **Create Variable**.

Step 4 Set the parameters listed in **Table 11-22**.

Figure 11-30 Creating an environment variable

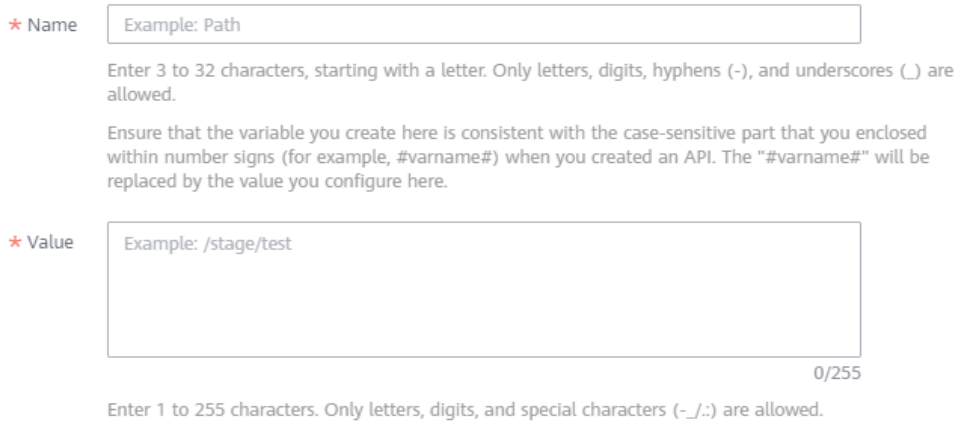


Table 11-22 Parameters for creating an environment variable

Parameter	Description
Name	Name of the variable you want to create. 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 5 Click **OK**.

NOTE

If a variable is not needed, click **Delete** in the row containing the variable to delete it.

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.

----End

Follow-Up Operations

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

Creating an Environment and Environment Variable by Calling an API

You can also create an environment and environment variable by calling an API provided by APIG. For details, see the following references:

Creating an Environment

Creating an Environment Variable

FAQ About Environment Variables

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

11.7.2 Deleting an Environment

Scenario

You can delete environments you no longer require.

Prerequisites

You have created an environment.

Procedure

Step 1 **Access the shared gateway console**.

Step 2 In the navigation pane, choose **API Publishing** > **Environments**.

Step 3 In the **Operation** column of the environment you want to delete, click **Delete**.

NOTE

You can delete an environment only if no APIs have been published in the environment.

Step 4 Click **Yes**.

----End

11.8 Signature Key Management

11.8.1 Creating and Using a Signature Key

Scenario

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.

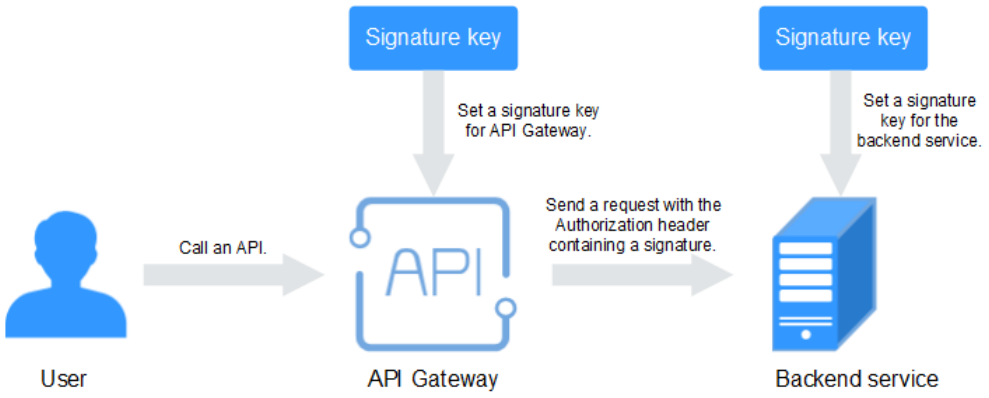
NOTE

Each API can only be bound with one signature key in a given environment, but each signature key can be bound to multiple APIs.

Procedure

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 (such as Java, Go, Python, JavaScript, C#, PHP, C++, C, and Android) to sign each request, and check whether the two signatures are consistent.

Figure 11-31 Signature key process flow



Creating a Signature Key

Step 1 Access the shared gateway console.

Step 2 In the navigation pane, choose **API Publishing** > **Signature Keys**.

Step 3 Click **Create Signature Key**.

Step 4 In the **Create Signature Key** dialog box, set the parameters listed in **Table 11-23**.

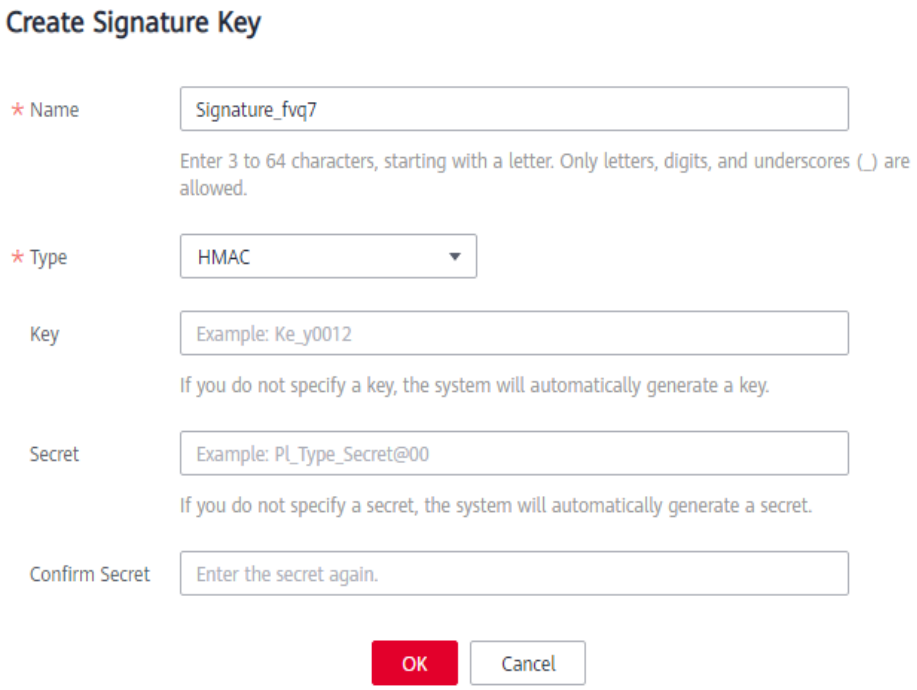


Table 11-23 Parameters for creating a signature key

Parameter	Description
Name	Signature key name.
Type	Type of the signature key. Select **HMAC** or **Basic**. This parameter is available only for dedicated gateways.
Key	Combined with **Secret** to form a signature key pair. • If you set **Type** to **HMAC**, enter the key of the key pair used for hash-based message authentication code (HMAC) authentication. • If you set **Type** to **Basic**, enter the username used for basic authentication.
Secret	Combined with **Key** to form a signature key pair. • If you set **Type** to **HMAC**, enter the secret of the key pair used for HMAC authentication. • If you set **Type** to **Basic**, enter the password used for basic authentication.
Confirm Secret	Enter the secret again.

Step 5 Click **OK**.

----End

Binding a Signature Key to an API

Step 1 In the navigation pane, choose **API Publishing** > **Signature Keys**.

Step 2 Bind a signature key to an API. You can use one of the following methods:

- In the **Operation** column of the signature key to be bound to an API, click **Bind to API**.
- Click the name of the target signature key.

Step 3 Click **Select API**.

Step 4 Specify an API group, environment, and API name keyword to search for the desired API.

Step 5 Select the API and click **OK**.

NOTE

If a signature key is no longer needed for an API, unbind it from the API.

----End

Verifying the Signing Result

Sign each backend request by following the instructions in **Signature Algorithm**, and check whether the backend signature is consistent with the signature in the **Authorization** header of the API request.

11.8.2 Deleting a Signature Key

Scenario

You can delete signature keys you no longer require.

Prerequisites

You have created a signature key.

Procedure

Step 1 **Access the shared gateway console**.

Step 2 In the navigation pane, choose **API Publishing** > **Signature Keys**.

Step 3 Delete a signature key. You can use one of the following methods:

- In the **Operation** column of the signature key you want to delete, click **Delete**.
- Click the name of the target signature key, and click **Delete** in the upper right corner of the displayed signature key details page.

NOTE

If the signature key has been bound to any APIs, unbind it and then delete it.

Step 4 Click **Yes**.

----End

11.9 VPC Channel Management

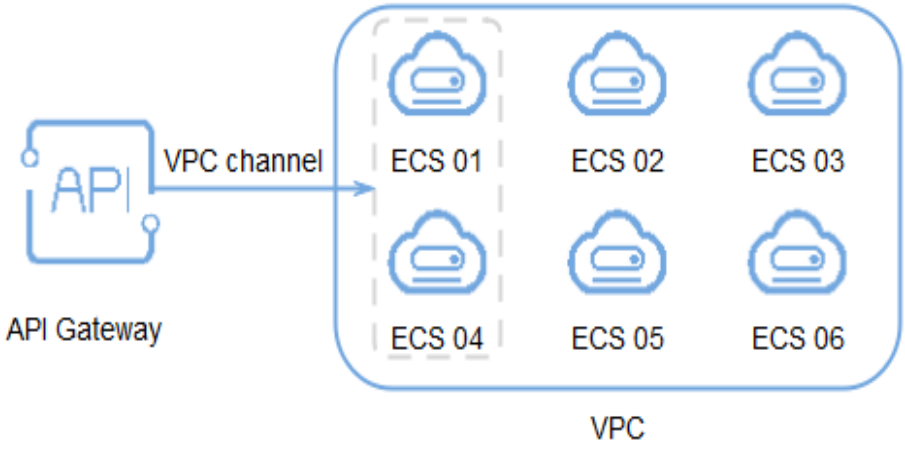
11.9.1 Creating a VPC Channel

Scenario

VPC channels allow services deployed in VPCs to be accessed through their subnets, lowering latency and balancing loads of backend services.

After creating a VPC channel, you can configure it for an API with an HTTP/HTTPS backend service. For example, six ECSs have been deployed in a VPC, and a VPC channel has been created to reach ECS 01 and ECS 04. APIG can access these two ECSs through the VPC channel.

Figure 11-32 Accessing ECSs in a VPC channel through APIG



NOTE

Prerequisites

- You have created a cloud server.
- You have the **VPC Administrator** permission.

Creating a Fast Channel

Step 1 Access the shared gateway console.

Step 2 In the navigation pane, choose **API Publishing** > **VPC Channels**.

Step 3 Click **Create VPC Channel**, and set the parameters listed in **Table 11-24**.

Figure 11-33 Creating a fast channel

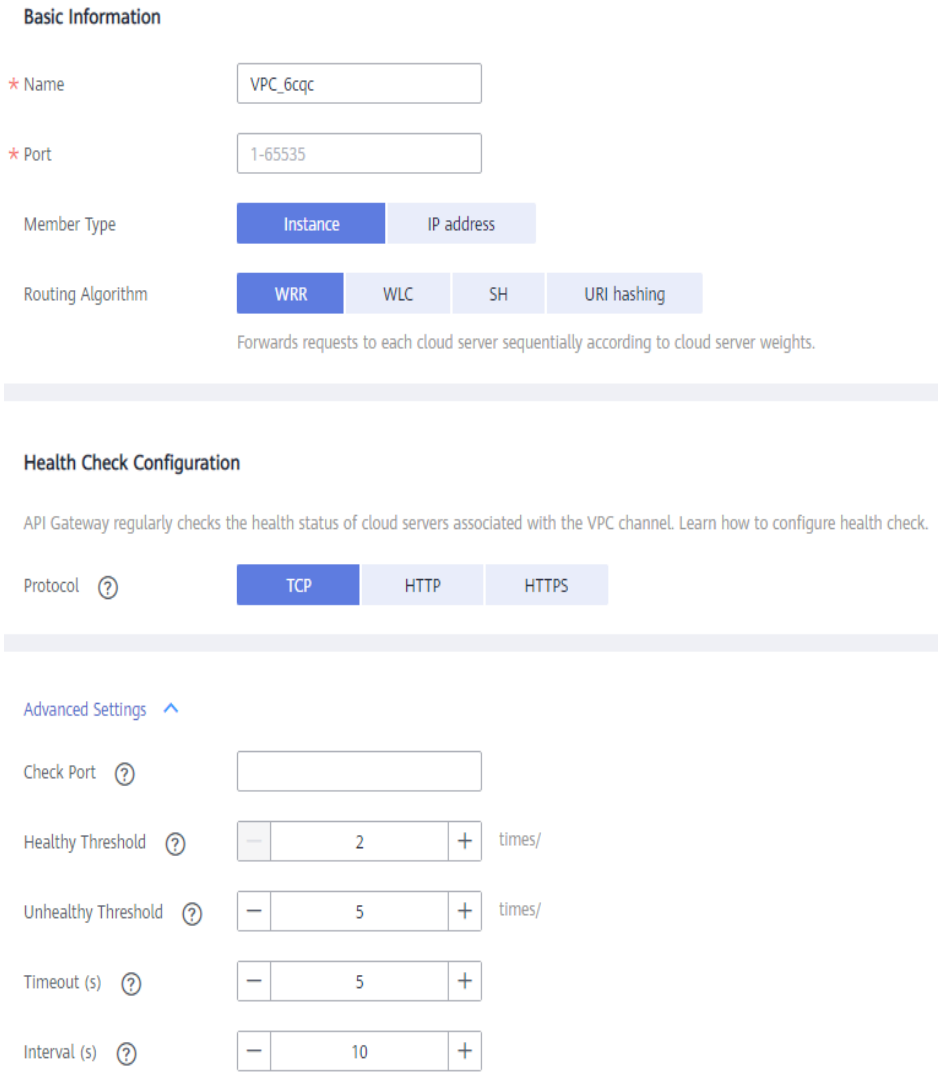


Table 11-24 Parameters for creating a VPC channel

Parameter	Description
Name	VPC channel name.
Port	The host port of the VPC channel, that is, the port of the backend service. Range: 1–65535.
Member Type	Select a method that you want to use to specify servers for the VPC channel. The member type is a one-time configuration and cannot be changed after you create the VPC channel. • **Instance**: Select cloud servers. • **IP address**: Specify cloud server IP addresses.

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**
Protocol	The protocol used to perform health checks on cloud servers associated with the VPC channel. Options: • TCP • HTTP • HTTPS Default value: **TCP**.
Path	The destination path for health checks. Set this parameter only when **Protocol** is not set to **TCP**.
Check Port	The destination port for health checks. By default, the port of the VPC channel will be used.
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	The HTTP codes used to check for a successful response from a target. Set this parameter only when **Protocol** is not set to **TCP**.

Step 4 Click **Next**.

Step 5 Click **Select Cloud Server**.

Step 6 Select the cloud servers you want to add, and click **OK**.

NOTE

When using the shared gateway, to ensure a successful health check and service availability, configure the security groups of the cloud servers to allow access from 100.125.0.0/16.

Step 7 Click **Finish**.

----End

Follow-Up Operations

Create an API for backend services deployed in a VPC to balance loads.

11.9.2 Deleting a VPC Channel

Scenario

You can delete VPC channels you no longer require.

NOTE

VPC channels that are currently in use by published APIs cannot be deleted.

Prerequisites

You have created a VPC channel.

Procedure

Step 1 **Access the shared gateway console**.

Step 2 In the navigation pane, choose **API Publishing** > **VPC Channels**.

Step 3 Delete a VPC channel. You can use one of the following methods:

- In the **Operation** column of the VPC channel you want to delete, click **Delete**.
- Click the name of the target VPC channel, and click **Delete** in the upper right corner of the displayed VPC channel details page.

Step 4 Click **Yes**.

----End

11.9.3 Editing Health Check Configurations

Scenario

You can modify the health check configurations of a VPC channel to meet service requirements.

Prerequisites

You have created a VPC channel.

Procedure

Step 1 Access the shared gateway console.

Step 2 In the navigation pane, choose **API Publishing** > **VPC Channels**.

Step 3 Click the name of the target VPC channel,

Step 4 Click the **Health Check** tab.

Step 5 Click **Edit Health Check**.

Step 6 In the **Edit Health Check Configuration** dialog box, modify the parameters listed in **Table 11-25**.

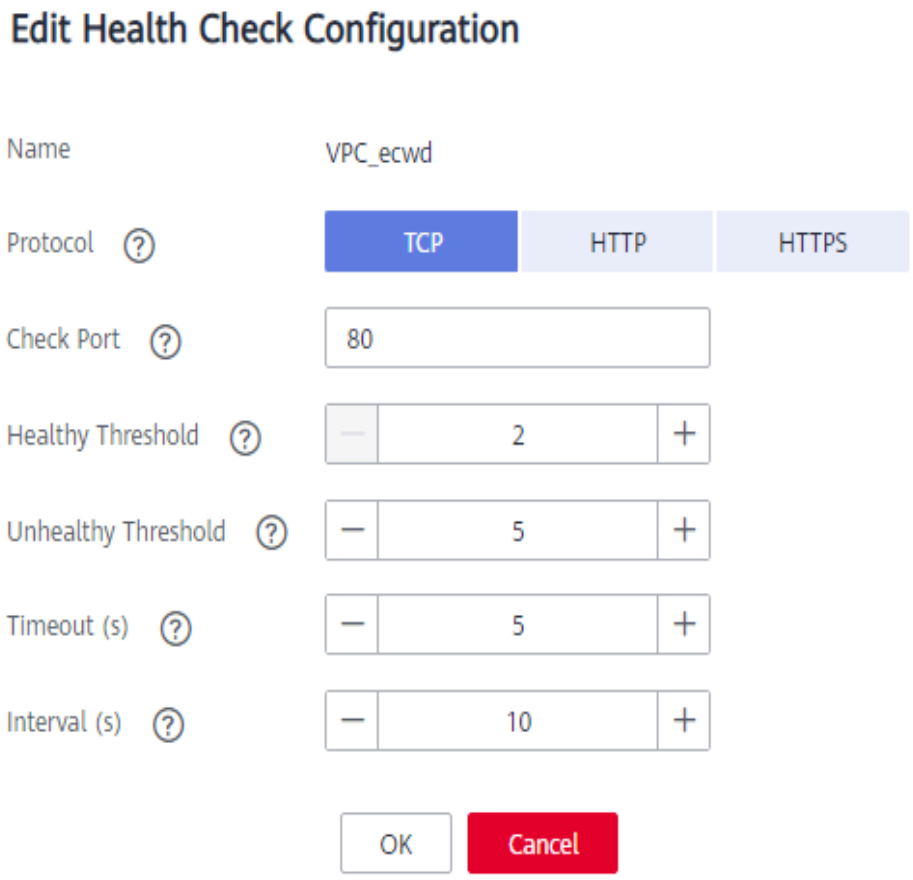


Table 11-25 Health check configurations

Parameter	Description
Protocol	The protocol used to perform health checks on cloud servers associated with the VPC channel. Options: • TCP • HTTP • HTTPS Default value: **TCP**.

Parameter	Description
Path	The destination path for health checks. Set this parameter only when **Protocol** is not set to **TCP**.
Check Port	The destination port for health checks. By default, the port of the VPC channel will be used.
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	The HTTP codes used to check for a successful response from a target. Set this parameter only when **Protocol** is not set to **TCP**.

Step 7 Click **OK**.

----End

11.9.4 Editing Cloud Server Configurations of a VPC Channel

Scenario

You can add or remove cloud servers and edit cloud server weights for VPC channels to meet service requirements.

Prerequisites

You have created a VPC channel.

Procedure

Step 1 **Access the shared gateway console**.

Step 2 In the navigation pane, choose **API Publishing** > **VPC Channels**.

Step 3 Click the name of the target VPC channel.

Step 4 Click the **Cloud Servers** tab.

Step 5 Add or remove cloud servers and edit cloud server weights.

- Adding cloud servers
  a. Click **Select Cloud Server**.
  b. Select the cloud servers you want to add, set cloud server weights, and click **OK**.

  NOTE

  To ensure a successful health check and service availability, configure the security groups of the backend cloud servers to allow access from 100.125.0.0/16.
- Removing cloud servers
  a. In the **Operation** column of the cloud servers you want to remove, click **Remove**.
  b. Click **Yes**.
- Editing the weight of a cloud server
  a. In the **Weight** column of the target cloud server, click .
  b. Change the weight and click .
- Editing the weights of multiple cloud servers
  a. Select the cloud servers to be edited, and click **Edit Weight**.
  b. Change the weights of the selected cloud servers, and click **OK**.

----End

11.10 Custom Authorizers

11.10.1 Creating a Custom Authorizer

Scenario

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 the backend authentication system.

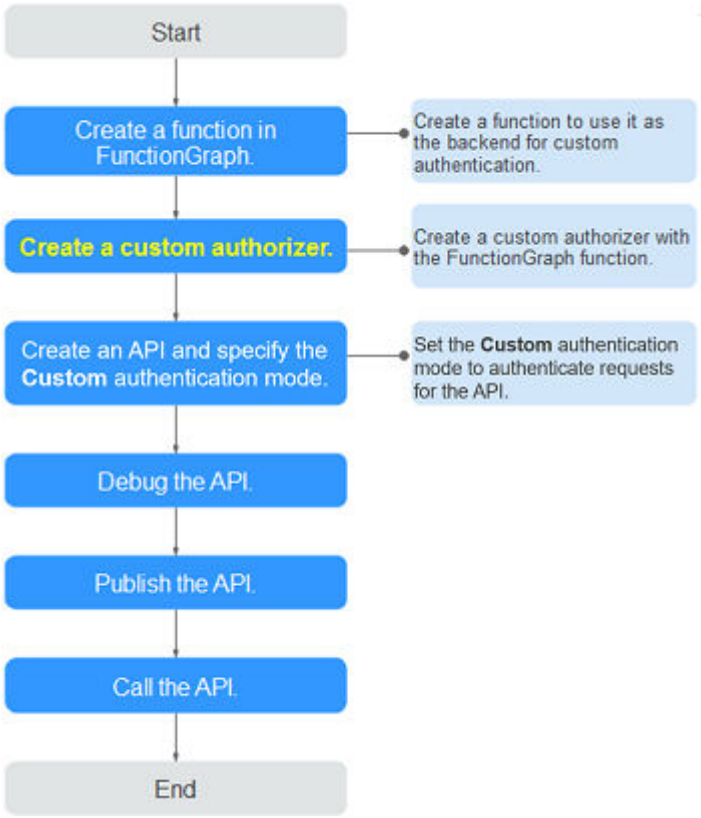
NOTE

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

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

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

Figure 11-34 Calling APIs through custom authentication



Prerequisites

- You have created a function in FunctionGraph.
- You have the **FunctionGraph Administrator** permission.

Procedure

Step 1 Access the shared gateway console.

Step 2 Choose **API Publishing** > **Custom Authorizers**, and click **Create Custom Authorizer**.

Step 3 Set the parameters listed in **Table 11-26**.

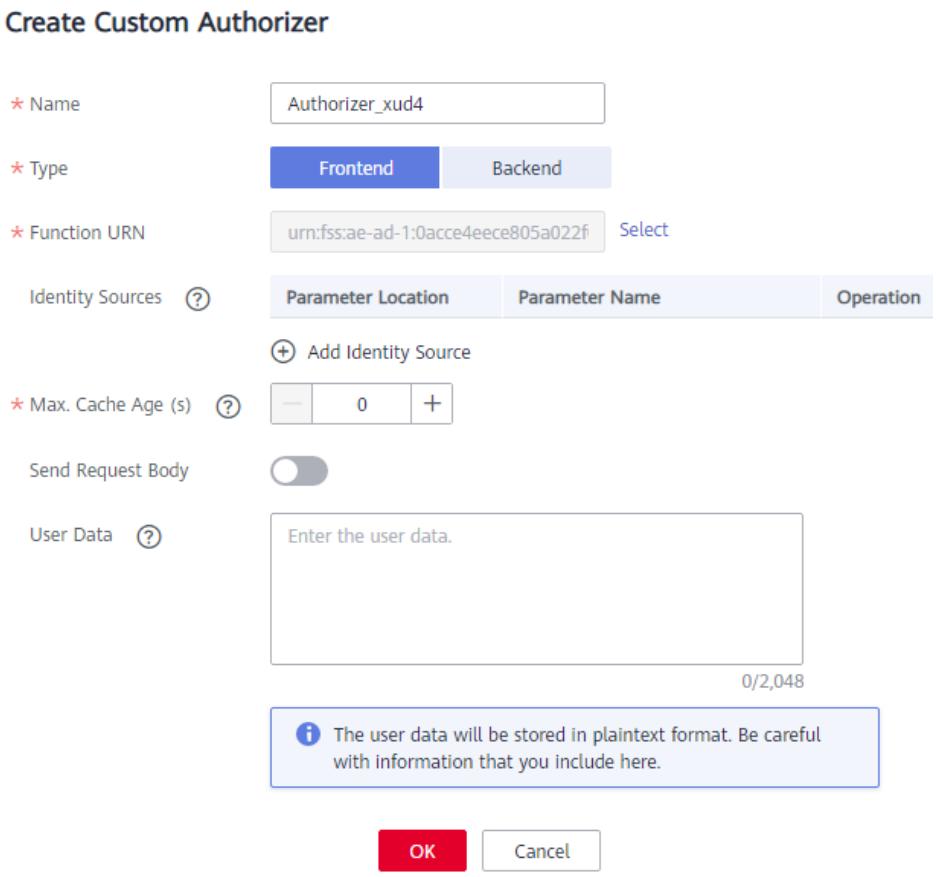


Table 11-26 Parameters for creating a custom authorizer

Parameter	Description
Name	Authorizer name.
Type	• **Frontend**: Authenticates access to APIs. • **Backend**: Authenticates access to backend services.
Function URN	Select a FunctionGraph function.
Identity Sources	Request parameters for authentication. You can add headers and query strings. Header names are case-insensitive. 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.
Max. Cache Age (s)	The time for caching authentication results. Value **0** means that authentication results will not be cached. The maximum value is **3600**.

Parameter	Description
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 4 Click **OK**.

----End

11.10.2 Deleting a Custom Authorizer

Scenario

You can delete custom authorizers you no longer require.

NOTE

- Custom authentication is implemented using FunctionGraph and not supported if FunctionGraph is unavailable in the selected region.
- Custom authorizers that have been configured for APIs cannot be deleted.

Prerequisites

You have **created a custom authorizer**.

Procedure

Step 1 **Access the shared gateway console**.

Step 2 Choose **API Publishing** > **Custom Authorizers**, and click **Delete** in the row containing the custom authorizer you want to delete.

Step 3 Click **Yes**.

----End

11.11 Monitoring

11.11.1 APIG 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

Shared gateway: SYS.APIG

Metrics

Table 11-27 Shared gateway metrics

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

Dimension

Table 11-28 Shared gateway monitoring dimension

Key	Value
api_id	API

11.11.2 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 Access the shared gateway console.

Step 2 In the navigation pane, choose **API Publishing** > **APIs**.

Step 3 Click the name of the target API.

Step 4 On the **Dashboard** tab page, click **View Metric** to access the Cloud Eye console. Then create an alarm rule. For details, see **Creating an Alarm Rule**.

----End

11.11.3 Viewing Metrics

Scenario

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

Prerequisites

You have created an API group and API.

Procedure

Step 1 Access the shared gateway console.

Step 2 In the navigation pane, choose **API Publishing** > **APIs**.

Step 3 Click the name of the target API.

API metrics are displayed on the **Dashboard** tab page.

Step 4 Click **View Metric** to view more metrics on the Cloud Eye console.

NOTE

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

----End

11.12 App Management

11.12.1 Creating an App and Obtaining Authorization

Scenario

For an API that uses app authentication, you can create an app and use the app and its ID and key pair (AppKey and AppSecret) to call the API. You can use an app to call an API only after you bind the app to the API. When you call the API, replace the key pair in the SDK with your own key pair so that APIG can authenticate your identity. For details about app authentication, see **Developer Guide**.

NOTE

- If the authentication mode of the target API has been set to **None** or **IAM**, you do not need to create apps to call this API.

Creating an App

Step 1 **Access the shared gateway console**.

Step 2 In the navigation pane, choose **API Calling** > **Apps**.

Step 3 Click **Create App**, and configure the app information.

Table 11-29 App information

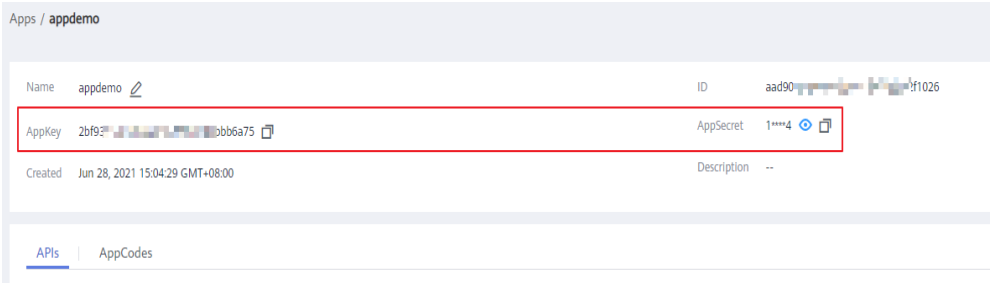
Parameter	Description
Name	App name.
Description	Description of the app.

Step 4 Click **OK**.

After the app is created, its name and ID are displayed in the app list.

Step 5 Click the app name, and view the AppKey and AppSecret on the app details page.

Figure 11-35 App details



----End

Binding an App to an API

Step 1 Access the shared gateway console.

Step 2 In the navigation pane, choose **API Calling** > **Apps**.

Step 3 Bind an app to an API. You can use one of the following methods:

- In the **Operation** column of the app, click **Bind to API**, and then click **Select API**.
- Click the name of the target app, and click **Select API**.

Step 4 Select an environment, select an API, and click **OK**.

After the binding is complete, you can view the API on the app details page.

NOTE

- Only APIs using app authentication can be bound with apps.
- An app can be bound to multiple APIs that use app authentication, and each such API can be bound with multiple apps.
- To debug an API to which the app is bound, click **Debug** in the row containing the API.

----End

Follow-Up Operations

You can call APIs using different authentication methods. For details, see **Calling APIs**.

11.12.2 Deleting an App

Scenario

You can delete apps you no longer require.

Prerequisites

You have created an app.

Procedure

Step 1 Access the shared gateway console.

Step 2 In the navigation pane, choose **API Calling** > **Apps**.

Step 3 Delete an app. You can use one of the following methods:

- In the **Operation** column of the app you want to delete, click **Delete**.
- Click the name of the target app, and click **Delete App** in the upper right corner of the displayed app details page.

NOTE

If the app has been bound to any APIs, you must unbind the app and then delete it.

Step 4 Click **Yes**.

----End

11.12.3 Resetting the AppSecret of an App

Scenario

You can reset the AppSecret of an app. The AppKey is unique and cannot be reset. When you reset the AppSecret, it becomes invalid and APIs bound to the app cannot be called. To enable API calls for that app again, you will need to update the AppSecret of the app you use.

Prerequisites

You have created an app.

Procedure

Step 1 Access the shared gateway console.

Step 2 In the navigation pane, choose **API Calling** > **Apps**.

Step 3 Click the name of the target app.

Step 4 In the upper right corner of the displayed app details page, click **Reset AppSecret**.

Step 5 Click **Yes**.

----End

11.12.4 Adding an AppCode for Simple Authentication

Scenario

AppCodes are identity credentials of an app used to call APIs in simple authentication mode. In this mode, the **X-Apig-AppCode** parameter (whose value is an AppCode on the app 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, AppKey and AppSecret can be used to sign and verify the API request. AppCode can also be used for simple authentication.

NOTE

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

Prerequisites

You have created an app.

Generating an AppCode

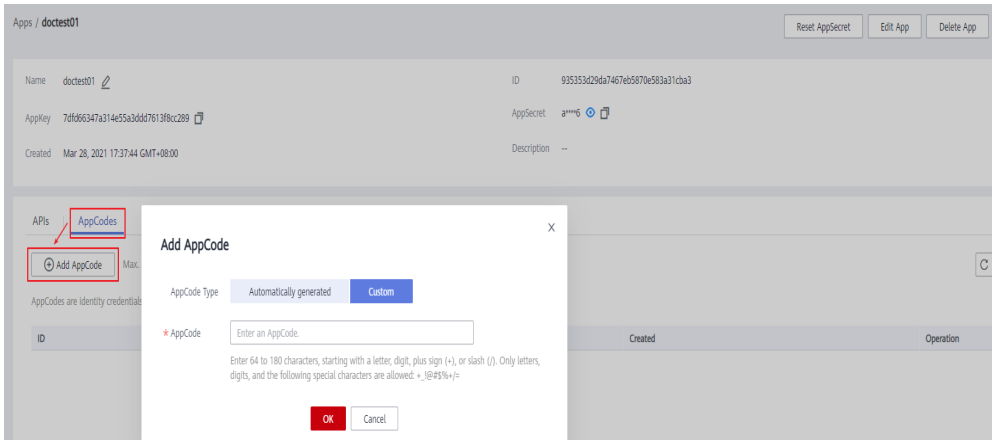
Step 1 Access the shared gateway console.

Step 2 In the navigation pane, choose **API Calling** > **Apps**.

Step 3 Click the name of the target app.

Step 4 Click the **AppCodes** tab.

Step 5 Click **Add AppCode** to generate an AppCode. It can be automatically generated or customized.



----End

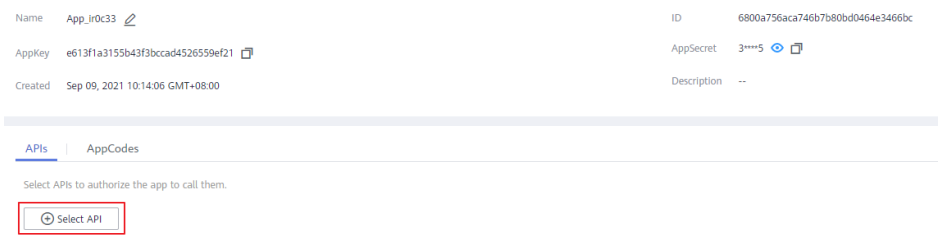
Using AppCode for Simple Authentication of API Requests

Step 1 When creating an API, set **Security Authentication** 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 an app 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

11.12.5 Viewing API Details

Scenario

You can view the details of an API to which an app has been bound.

Prerequisites

- You have created an app.
- The app has been bound to an API.

Procedure

Step 1 **Access the shared gateway console**.

Step 2 In the navigation pane, choose **API Calling** > **Apps**.

Step 3 Click the name of the target app.

Step 4 Click the name of the target API to view its details.

----End

11.13 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. For details about IAM authentication, see **Using IAM Authentication to Call APIs**.

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

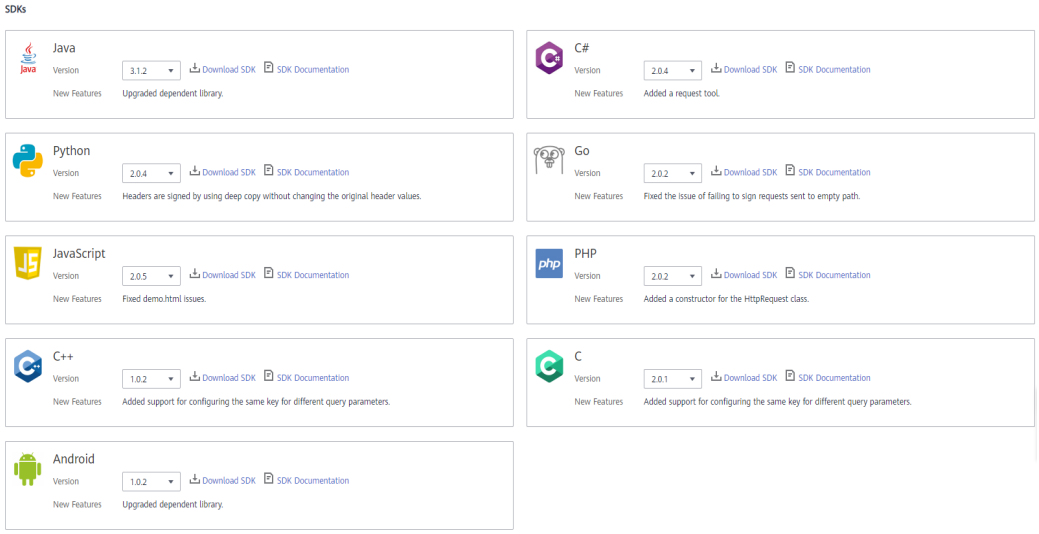
Step 1 **Access the shared gateway console** and click  .

Step 2 Choose **Help Center**.

Step 3 Click **SDK Process Flow**.

Step 4 Click **Download SDK** of the desired language.

To view the support guide, click **SDK Documentation**.



----End

11.14 Purchased APIs

Scenario

In the shared gateway, you can view the purchased APIs and debug the APIs to check whether they are running properly.

Purchased APIs must be called using app authentication.

Prerequisites

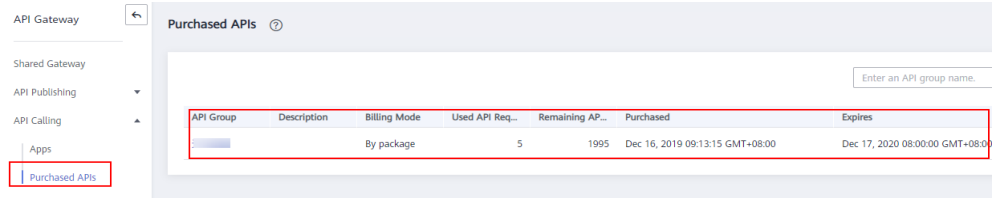
You have purchased APIs through KooGallery.

Procedure

Step 1 **Access the shared gateway console**.

Step 2 In the navigation pane, choose **API Calling** > **Purchased APIs**.

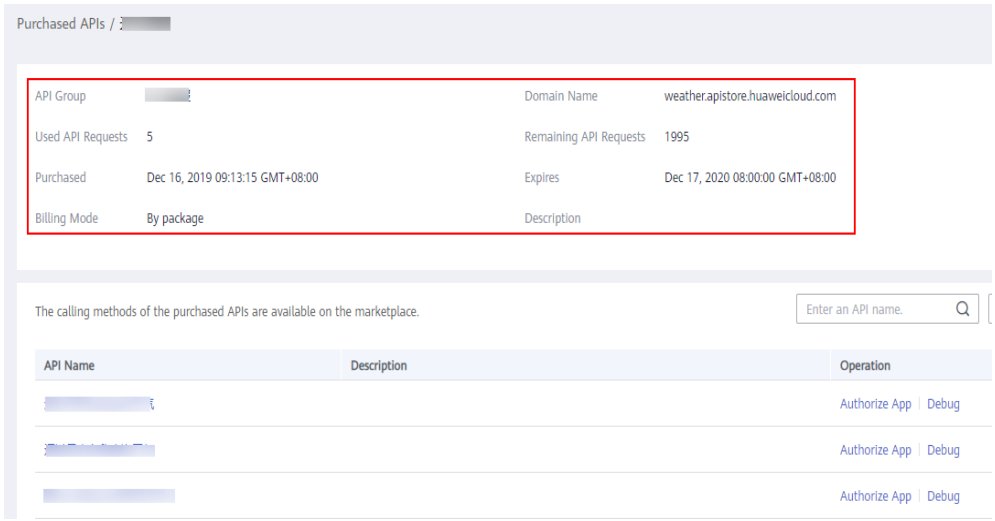
Figure 11-36 Purchased API group



Step 3 Click the name of the target API group.

Details of the API group and purchased APIs under the group are displayed.

Figure 11-37 API group details



Step 4 In the **Operation** column of the desired API, click **Debug**.

Step 5 On the left side, set the API request parameters listed in **Table 11-30**. On the right side, view the API request and response information after you click **Send Request**.

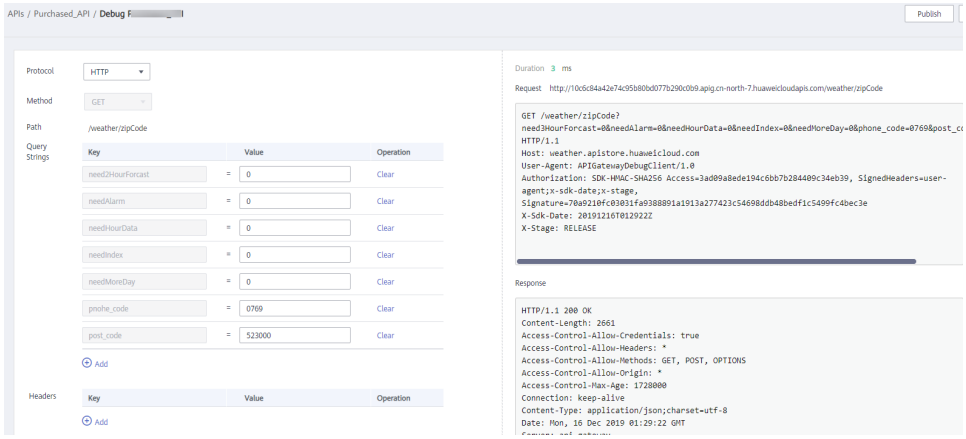
Table 11-30 Parameters for debugging an API

Parameter	Description
Protocol	You can modify this parameter only if you have set **Protocol** to **HTTP&HTTPS** for the API.
Method	You can modify this parameter only if you have set **Method** to **ANY** for the API.
Suffix	You can modify this parameter only if you have set **Matching** to **Prefix match** for the API.
Path Parameters	You can modify this parameter only if the value of **Path** contains braces ({}).
Headers	HTTP headers and values.
Query Strings	Query string parameters and values.

Parameter	Description
Body	You can modify this parameter only if you have set **Method** to **PATCH**, **POST**, or **PUT** for the API.

Step 6 After setting request parameters, click **Send Request**.

The **Response** section displays the response of the API request.



Step 7 You can send more requests with different parameters and values to verify the API.

----End

11.15 Calling Published APIs

11.15.1 Calling APIs

Obtaining APIs and Documentation

Before calling APIs, obtain the request information from the API provider, including the access domain name, protocol, method, path, and request parameters.

Obtain APIs: from your company or from a partner

Obtain related documentation

- For APIs obtained from Huawei Cloud, obtain documentation from the **Help Center**.

The authentication information to be obtained varies with the API authentication mode.

- App authentication:
  - Signature authentication: Obtain the key and secret (or client AppKey and AppSecret) of the app authorized for the API from the API provider as well as the SDK for calling the API.
  - Simple authentication: Obtain the AppCode of the app authorized for the API from the API provider.

  - Other authentication modes: Obtain the key and secret (or client AppKey and AppSecret) of the app authorized for the API from the API provider.
- **IAM authentication**: The account credential (token or AK/SK obtained with the account and password) obtained on the cloud service platform is used for authentication. If the AK/SK is used for authentication, you also need to obtain the SDK from the API provider for calling the API.
- Custom authentication: Obtain the custom authentication information to be carried in the request parameters from the API provider.
- None: No authentication information is required.

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.

Step 1 Set the request path.

Scenario	Request Parameter Configuration
Calling an API with a domain name	Call the API using **the subdomain name allocated to the API group or a domain name bound to the group**. No additional configuration is required.
Calling an API in a non-DEFAULT group with an IP address	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**.

Step 2 Set the authentication parameters.

Authentication Mode	Request Parameter Configuration
App authentication (with a signature)	Use the SDK to sign API requests. For details, see **Calling APIs Through App Authentication**.
App authentication (through simple authentication)	Add the header parameter **X-Apig-AppCode** and set the parameter value to the AppCode obtained in **Obtaining APIs and Documentation**. For details, see **Getting Started**.
IAM authentication (with a token)	Obtain a token from the cloud platform and carry the token in API requests for authentication. For details, see **Token Authentication**.
IAM authentication (with AK/SK)	Use an SDK to sign API requests. For details, see **AK/SK Authentication**.

Authentication Mode	Request Parameter Configuration
Custom authentication	Carry authentication information in API request parameters for authentication.
None	Call APIs without authentication.

----End

11.15.2 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	App 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 app.

Response Header	Description	Remarks
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**.

11.15.3 Error Codes

Table 11-31 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 **Error Codes**.

NOTE

- For details about the error codes that may occur when you manage APIs, see **Error Codes**.
- 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 11-31 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 registered 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.
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.

Error Code	Error Message	HTTP Status Code	Description	Solution
APIG.0106	Orchestration error.	400	An orchestration error occurred.	Check whether the frontend and backend parameters of the API are correct.
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.
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.

Error Code	Error Message	HTTP Status Code	Description	Solution
APIG.0203	Backend timeout.	504	The backend service has timed out.	Increase the timeout of the backend service or shorten its 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.0301	Incorrect IAM authentication information.	401	The IAM authentication details are incorrect.	Check the token by referring to **Common Errors Related to IAM Authentication Information**.
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.

Error Code	Error Message	HTTP Status Code	Description	Solution
APIG.0303	Incorrect app authentication information.	401	The app authentication details are incorrect.	Perform the following checks for app authentication: • Check whether the request method, path, query parameters, and request body are consistent with those used for signing. • Check whether the client time is correct. Check whether the signing code is correct by referring to **Calling APIs Through App Authentication**. In the case of AppCode-based simple authentication, check whether the request contains the **X-Apig-AppCode** header.
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.

Error Code	Error Message	HTTP Status Code	Description	Solution
APIG.0307	The token must be updated.	401	The token needs to be updated.	Obtain a new token from IAM.
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 subdomain requests per day is reached, bind an independent domain name to the API.
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.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.0501	The app quota has been used up.	405	The app quota has been reached.	Increase the app quota.

Error Code	Error Message	HTTP Status Code	Description	Solution
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.
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 tenant details cannot be obtained.	Contact technical support.
APIG.0609	The VPC backend does not exist.	500	The VPC backend service cannot be found.	Contact technical support.

Error Code	Error Message	HTTP Status Code	Description	Solution
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.
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.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.1009	AppKey or AppSecret is invalid	400	The AppKey or AppSecret is invalid.	Check whether the AppKey or AppSecret in the request is correct.

11.16 Key Operations Recorded by CTS

11.16.1 APIG operations that can be 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)**.

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 11-32 APIG operations that can be recorded by CTS

Operation	Resource Type	Trace Name
Creating an API group	ApiGroup	createApiGroup
Deleting an API group	ApiGroup	deleteApiGroup
Updating an API group	ApiGroup	updateApiGroup
Binding a domain name	ApiGroup	createDomainBinding
Change minimum TLS version	ApiGroup	modifySecureTransmis-sion
Unbinding a domain name	ApiGroup	relieveDomainBinding
Adding a domain certificate	ApiGroup	addDomainCertificate
Deleting a domain certificate	ApiGroup	deleteDomainCertificate
Creating an API	Api	createApi
Deleting an API	Api	deleteApi
Deleting multiple APIs	Api	batchDeleteApi
Updating an API	Api	updateApi
Publishing an API	Api	publishApi
Taking an API offline	Api	offlineApi

Operation	Resource Type	Trace Name
Publishing multiple APIs or taking APIs offline	Api	batchPublishOrOfflineApi
Switching API versions	Api	switchApiVersion
Taking an API version offline	Api	offlineApiByVersion
Debugging an API	Api	debugApi
Creating an environment	Environment	createEnvironment
Deleting an environment	Environment	deleteEnvironment
Updating an environment	Environment	updateEnvironment
Creating an environment variable	EnvVariable	createEnvVariable
Updating an environment variable	EnvVariable	updateEnvVariable
Deleting an environment variable	EnvVariable	deleteEnvVariable
Creating an app	App	createApp
Deleting an app	App	deleteApp
Updating an application	App	updateApp
Resetting AppSecret	App	resetAppSecret
Binding a client to an API	AppAuth	grantAuth
Unbinding a client from an API	AppAuth	relieveAuth
Creating a signature key	Signature	createSignature
Deleting a signature key	Signature	deleteSignature
Updating a signature key	Signature	updateSignature
Binding a signature key	SignatureBinding	createSignatureBinding
Unbinding a signature key	SignatureBinding	relieveSignatureBinding
Creating an access control policy	Acl	createAcl
Deleting an access control policy	Acl	deleteAcl

Operation	Resource Type	Trace Name
Deleting access control policies	Acl	batchDeleteAcl
Updating an access control policy	Acl	updateAcl
Creating an access control blacklist	Acl	addAclValue
Deleting an access control blacklist	Acl	deleteAclValue
Binding an access control policy to an API	AclBinding	createAclBinding
Unbinding an access control policy from an API	AclBinding	relieveAclBinding
Unbinding multiple access control policies from APIs	AclBinding	batchRelieveAclBinding
Creating a request throttling policy	Throttle	createThrottle
Deleting a request throttling policy	Throttle	deleteThrottle
Deleting multiple request throttling policies	Throttle	batchDeleteThrottle
Updating a requesting throttling policy	Throttle	updateThrottle
Binding a request throttling policy	ThrottleBinding	createThrottleBinding
Unbinding a request throttling policy	ThrottleBinding	relieveThrottleBinding
Unbinding multiple request throttling policies	ThrottleBinding	batchRelieveThrottle-Binding
Creating an excluded request throttling configuration	ThrottleSpecial	createSpecialThrottle
Deleting an excluded request throttling configuration	ThrottleSpecial	deleteSpecialThrottle

Operation	Resource Type	Trace Name
Updating an excluded request throttling configuration	ThrottleSpecial	updateSpecialThrottle
Creating a load balance channel	Vpc	createVpc
Deleting a load balance channel	Vpc	deleteVpc
Updating a load balance channel	Vpc	updateVpc
Adding members to a load balance channel	Vpc	addVpcMember
Deleting members from a load balance channel	Vpc	deleteVpcMember
Exporting an API	Swagger	swaggerExportApi
Exporting multiple APIs	Swagger	swaggerExportApiList
Exporting all APIs in a group	Swagger	swaggerExportApi-ByGroup
Importing APIs to a new group	Swagger	swaggerImportApiTo-NewGroup
Importing APIs to an existing group	Swagger	swaggerImportApiToEx-istGroup
Exporting all custom backends	Swagger	SwaggerExportLdApi
Importing custom backends	Swagger	SwaggerImportLdApi
Creating a custom authorizer	Authorizer	createAuthorizer
Deleting a custom authorizer	Authorizer	deleteAuthorizer
Updating a custom authorizer	Authorizer	updateAuthorizer

Disabling CTS

Disable CTS by following the procedure in **Deleting a Tracker**.

11.16.2 Viewing Audit Logs

Query audit logs by following the procedure in **Querying Real-Time Traces**.

Figure 11-38 Viewing logs