

API Gateway

User Guide for Providers

User Guide for Providers

Overview

API Gateway provides high-performance and highly available API hosting service to help users to publish or access to the APIs on Alibaba Cloud products such as ECS and Container Service. It manages the entire API lifecycle from release and management to maintenance. You can quickly open data or services at low costs and risks through simple operations.

API Gateway provides the following features:

API management

You can manage the lifecycle of an API, including creation, testing, release, deprecation, and version switching.

Easy data conversion

You can configure a mapping rule to convert the calling request into the format required by the backend.

Presetting of request verification

You can preset the verification of the parameter type and values (range, enumeration, regular expression, and JSON Schema) for gateway to preclude the invalid requests, reduce the utilization rate of your backend.

Flexible throttling

You can set throttling for APIs, users, and APPs by minute, hour, or day.

In addition, you can also specialize some users or APPs with the independent throttling.

Easy security protection

API Gateway supports AppKey authentication and HMAC (SHA-1,SHA-256) signature.

API Gateway supports SSL/TSL encryption and uses Alibaba Cloud Security to prevent viruses and attacks.

Comprehensive monitoring and warning

API Gateway provides visualized API monitoring in real time, including the calling traffic, calling method, response time, and error rate, and supports query of historical records for comprehensive analysis. You can also configure and subscribe to the warning method (SMS or email) to check the API running status in real time.

Lower cost of publication

API Gateway automatically generates API documentation and SDKs (service end and mobile end), reducing the cost of publication of API.

Create an API

When you create an API, you must enter the basic information about the API, and define the API request information, the API backend service, and response information. Then, debug the API and set the security configuration. After testing the API, if it works properly, you can publish it to the **Release** environment for your users.

Define an API

On the **API List** page in the API Gateway console, click **Create API** to enter the API creation and definition process.

Basic Information of API

Basic API information includes the API group, API name, security certification method, visibility type, and description.

- **Group:** APIs are managed by group. Before creating an API, you must first create a group (for more information about API groups, see **Open an API**).
- **API Name:** It is the name of an API and also an unique identifier within its group.

Security Certification: This method is used to authenticate API requests. Currently, the four methods available are **Alibaba Cloud App**, **OpenID Connect**, **OpenID Connect & Alibaba Cloud App**, and **No Certification**.

- **Alibaba Cloud App:** When the requestor calls this API, they must pass the identity authentication of this app.
- **OpenID Connect:** This is a lightweight standard based on OAuth 2.0, which provides a framework for identity interaction through RESTful APIs. You can use OpenID Connect to connect seamlessly to your own account system. For a detailed introduction, see [OpenID Connect](#).
- **OpenID Connect & Alibaba Cloud App:** Both OpenID Connect and Alibaba Cloud App authentication
- **No Certification:** Any person who knows this API's request definitions can initiate a request. The gateway does not verify their identities, but directly forwards the requests to your backend service. (We strongly recommend not to use this method.)

Visibility: **Public** or **Private**.

- **Public APIs:** All the users can view a certain sections of the API's information on the **Published API** page of the API Gateway console.
- **Private APIs:** You must manually grant authorization to the user, if the user wants to call your private API.

- **Description:** A description of the functions of an API.

Define API request

This part defines how users can send requests to your API, including the relevant protocols, request paths, HTTP methods, request modes, and input parameter definitions.

- **Protocol:** Supports HTTP and HTTPS.
- **Request Path:** The path refers to the API request path for the corresponding service host. The request path can be different from the actual backend service path. You can write any valid and semantically-clear path for users. You can configure dynamic parameters in the request path that require users to input parameters in the Path field. At the same time, the backend service receives parameter from the Path, which are mapped to Query, Header and other locations. In **Published API to API Gateway**, you can find detailed examples and operation screenshots.
- **HTTP Method:** Supports standard HTTP methods. You can select PUT, GET, POST, DELETE, PATCH, or HEAD.
- **Request Mode:** API Gateway used this mode to process input parameters, **Request Parameter Mapping** or **Request Parameter Passthrough**.
 - **Request Parameter Mapping:** When API Gateway receives a request for your API, it uses mapping relationships to convert the request to the format required by your backend service. Request Parameter Mapping mode features:
 - Definition method: When defining this API, you must add the frontend and backend parameter mapping relationships.
 - Scenarios:

- For the same interface, define different APIs in API Gateway to provide differentiated services to users.
 - Use API Gateway to standardize legacy system interfaces.
 - Functions:
 - You can configure full frontend to backend mapping, that is, parameter shuffling. For example, you can require API users enter parameters in the **Query** field and set the backend to receive the information in the **Header** field. Supports parameter name conversion and parameter location conversion.
 - You can define parameter verification rules to pre-verify the request parameters, and to reduce the volume of invalid parameters that are processed in the backend. Supports length verification, parameter value verification, parameter regular expression verification, and parameter JSON schema verification.
 - **Request Parameter Passthrough:** After API Gateway receives an API request, it does not process the request, but directly forwards it to the backend service. In this mode:
 - You cannot implement parameter verification.
 - You cannot generate detailed API calling documentation.
 - Automatically-generated SDKs do not include request input parameters.
- **Input Parameter Definition:** You can define API request input parameters in this section, including parameter names, parameter locations, types, required or not, their default values, examples, and descriptions. **In Request Parameter Passthrough mode, users do not have to enter parameters.**
- **Parameter Name:** The parameter name displayed to users.
 - **Parameter Location:** The location of the parameter mentioned in the request that includes Head, Query, and Parameter Path.
- Note:** If you configure dynamic parameters in Path, the parameter location also defined as Parameter Path.
- **Type:** The field type; optional values: String, Int, Long, Float, Double, and Boolean.
 - **Required:** Indicates whether this parameter is required. When set to **Yes**, API Gateway verifies that user requests contain this parameter. Requests without this parameter are rejected.
 - **Default Value:** This option is applied when **Required** is set to **No**. If a user's request does not contain the corresponding parameter, API Gateway automatically adds the default value before sending the request to the backend service.
 - **Example:** An example of defining parameters.
 - **Description:** Provides a brief description about the parameter and also mentions points to consider while using it.
 - **Parameter Verification Rules:** Click **More** to configure verification rules for the parameter value, including string length, the minimum and maximum values,

enumeration, regular expressions, JSON schema, and other attributes. API Gateway uses the verification rules to perform preliminary inspections on requests. If an input parameter is invalid, the request is not sent to the backend service. This is to reduce the backend processing load.

Define API backend service

This part mainly defines parameter mapping between the frontend and backend. This is the API backend service configuration, including the backend service address, backend path, backend timeout, parameter mapping, constant parameters, and system parameters. After user requests reach API Gateway, it maps the received requests according to your backend configuration, to the format required by the backend service before the requests are forwarded to the backend service.

- **Backend Service Type:** Currently supports HTTP/HTTPS and Function Compute.
 - HTTP/HTTPS: Select this option if your service is an HTTP/HTTPS service.

NOTE: If you have an HTTPS service, the backend service must have an SSL certificate.
 - Function Compute: If you select Function Compute for the backend service, you must first create a function in the Function Compute console, enter the function's service name and function name, and obtain Function Compute's role Arn.
- **Backend VPC Access:** When your backend service is in a VPC network, you must select **Enable**. For the usage method, see [Open an API in a VPC environment](#).
- **Backend Service Address:** The host of the backend service. This can be a domain name and in an 'http(s)://host:port' format. This value must begin with "http://" or "https://".
- **Backend Request Path:** This path is the actual request path of your API service on the backend server. If your backend path receives dynamic parameters, specify a particular location and name of the parameter a caller must enter. This declares the corresponding mapping relationship.
- **Backend Timeout:** This is the maximum length of time during which an API must receive a response from the backend service of the called API. This period starts when API Gateway sends a request to the backend service and ends when API Gateway receives a response result from the backend service. Units: milliseconds. This value cannot exceed 30 seconds. If the response time exceeds this value, API Gateway abandons the request and returns the corresponding error message to the user.
- **Constant Parameters:** You can configure constant parameters. These parameters are invisible to the users. However, when requests pass through API Gateway, it adds these parameters to the specified locations in the requests before forwarding the requests to the backend service. This is used to address certain business needs of the backend service. For example, if you require that each request sent by API Gateway to the backend service carry the keyword **aligateway**, you can configure **aligateway** as a constant parameter and specify the location where it is received.

System Parameters: These are API Gateway system parameters. By default, system parameters are not transmitted to you. However, to obtain system parameters, configure their locations and names in the API. The specific content is shown in the following table.

Parameter	Description
CaClientIp	IP address of the client sending the request
CaDomain	Domain name sending the request
CaRequestHandleTime	Request time (GMT)
CaAppId	The ID of the request app
CaRequestId	RequestId
CaApiName	API name
CaHttpSchema	Protocol used by the user to call the API: HTTP or HTTPS.
CaProxy	Proxy (AliCloudApiGateway)

Note: You must make sure that the names of all parameters are globally unique, including the dynamic parameters in Path, Headers parameters, Query parameters, Body parameters (non-binary), constant parameters, and system parameters. If you have a parameter called **name** in the Headers and Query fields at the same time, the system reports an error.

Part 4: Define response

Enter the returned ContentType, response example, failed response example, and error code definitions.

Debug an API

After you define an API successfully, you can debug it on the API debugging page to verify its correctness and usability.

After you create and define an API, you can test whether the created API is usable and the request chain is correct.

1. Click the API name or the **Manage** button to go to the **API Definition** page.
2. Click **Debug API** from the left-side navigation pane.
3. Enter the request parameters and click **Send Request**. The returned results are displayed on the right side of the page. If it returns a successful result, it indicates that the API can be used. If a 4XX or 5XX error code is returned, it indicates that the request has encountered an error. For more information, see [How to obtain the error message and Error code table](#).

Subsequent steps

After completing the API definition and preliminary debugging, you have finished creating an API. You can publish the API in the **Test**, **Pre**, or **Release** environments for ongoing debugging or for the other users to use. You can also bind a **Signature** key to the API and set **Throttling** and other security configurations.

Enable API services

Enable API services

This section provides information you must understand for the API group and domain name before you enable API services.

API group

An API group is the management unit of APIs. You must create a group before creating an API. The group consists of four attributes: name, description, region, and domain name. Note that:

- The group region is fixed once selected.

- Each account can have up to 50 API groups and each API group can have up to 200 APIs.

- When you create a group, the system assigns the group a second-level domain name to test your API. To enable the API service, you must bind the group to an independent domain name filed on Alibaba Cloud and resolve the CNAME of the independent domain name to the second-level domain name of the group. Up to five independent domain names can be bound to a group.

Domain name and certificate

API Gateway locates the unique API group through the domain name, and the unique API through the Path+HTTPMethod. Before enabling API services, you must know the second-level domain name and independent domain name as follows:

- The unique and fixed second-level domain name is assigned by the system during group creation. By default, a second-level domain name is used to call the API only in the test

environment under a small amount of traffic.

An independent domain name is used for enabling API services. You can bind up to five independent domain names to a group. When configuring independent domain names, pay attention to the following points:

Resolve the CNAME of an independent domain name to the API second-level domain name of the group before binding the API group and domain name.

Verify the domain name within one day. Otherwise, the unprocessed binding request is automatically withdrawn by the system.

If a domain name is already bound to another group, resolve the domain name to the second-level domain name of the to-be-bound group before binding. Otherwise, the binding fails.

If your API supports the HTTPS protocol, you must upload the SSL certificate of the domain name by entering the parameters on the **Group Details** page, including the name, content, and private key.

Test, production, and authorization

To test or enable the API, authorization is indispensable. Authorization means granting an app the permission to call an API. Note that:

- You can authorize the created app and access the second-level domain name to call the API.
- You can authorize the apps of customers to access the independent domain name to call your API service.
- Only an authorized app can call the API.

Now you have successfully enabled your API service. From creating the API to enabling it, you can create, modify, delete, view, test, release, remove, authorize, and revoke the authorization of an API. You can also view the release history and switch the version.

Manage an API

API definitions refer to the definitions related to the API request structure when you create an API. You can view, edit, delete, create, or copy an API definition on the console. Pay attention to the following points when you are working with API definitions:

1. Editing the definition of a released API does not affect the definition in the production environment unless you release and synchronize it to the production environment.
2. It is not allowed to directly delete the API definition. Deprecate the API definition before deleting it.
3. You can copy the definition from the test/production environment to overwrite the latest definition, and then, if needed, click **Edit** to modify the definition.

API release management

You can release or deprecate an API in a test or production environment with the following attentions:

1. You can access the second-level domain name or independent domain name to call the API that is released to the test or production environment.
2. The latest released version of an API overwrites the preceding version in the test/production environment and takes effect in real time.
3. When you deprecate an API in the test/production environment, the binding policy, keys, app, and authorization persists are automatically deprecated unless the API is released to production again. To revoke this relationship, you must delete it.

API authorization management

You can establish or revoke the authorization relationship between an API and an app. API Gateway verifies the permission relationship. During authorization, pay attention to the following points:

1. You can authorize one or more APIs to one or more apps. We recommend that you do not operate APIs in multiple groups at the same time during batch operation.
2. During batch operation, select an API and related environment. For example, if an API has been released to both the test and production environments, but only the test environment is chosen, only the API in the test environment is authorized.
3. You can locate an app based on the AppID or Alibaba Mail account provided by the customer.
4. When you need to revoke the authorization for an app under an API, you can view the API authorization list and delete the app from the list.

Release history and version switching

You can view the release history of each of you APIs, including the version number, notes, test/production, and time of each release.

When viewing the release history, you can select a version and switch to it. The new version directly overwrites the previous one and takes effect in real time.

plugin

Plugin Overview

In the new API Gateway version released in 2019, the original throttling, IP address-based access control, backend signature, and JWT (OpenId Connect) features are all integrated into the plug-in system. Cross-origin resource access (CORS) and caching are two new features added in this version. You can implement these new features by configuring plug-ins. In the future, more plug-ins will be added to API Gateway, and the API editing interface will be cleaned up.

Plug-ins are available only in the following regions. They will become available in other regions soon.

- UK (London)
- UAE (Dubai)
- US (Virginia)
- US West 1 (Silicon valley)

Plug-in limits

- You can only bind plug-ins of the same type to an API.
- Plug-ins are region-specific. You can bind a plug-in to an API in the same region as it. You can create a maximum of 500 plug-ins.
- Plug-ins and APIs are managed separately. A plug-in takes effect on an API only after the plug-in is bound to that API in the specified environment.
- You can bind plug-ins only to published APIs.
- The binding, unbinding, or update operations on plug-ins take effect immediately on their bound APIs. You do not need to republish these APIs. Some APIs may affect your business, we recommend that you test the plug-ins on a test API first.
- The operation of unpublishing an API does not affect its binding relationship with any plug-ins. The bound plug-ins take effect on the API again when the API is republished.
- Plug-ins that are bound to published or unpublished APIs cannot be deleted.

Supported plug-ins

API Gateway supports the following types of plug-ins. For more information about each type, follow the link provided.

- Throttling
- IP address-based access control
- Backend signature
- JWT (OpenID Connect)
- CORS
- Caching

Quick start

- Log on to the API Gateway console and create a plug-in on the Plugin List page.



- On the Plugin List page, bind the plug-in you created to a published API.



- The plug-in takes effect immediately after it is bound to the API.

Developer Guide

Thank you for choosing Alibaba Cloud API Gateway. You can use the APIs described in this document to perform relevant operations on API Gateway.

- CreatePlugin: creates a plug-in.
- ModifyPlugin: modifies a plug-in.
- DeletePlugin: deletes a plug-in.
- DescribePlugins: queries details of plug-ins.

- AttachPlugin: binds the specified plug-in to an API.
- DetachPlugin: unbinds the specified plug-in from an API.
- DescribePluginApis: checks the APIs to which a specified plug-in is bound.
- DescribePluginsByApi: checks the plug-ins bound to a specified API that is running in the specified environment.

Throttling

Throttling is now integrated into the plug-in system. The original throttling interface and console are still in use. Throttling policies and throttling plug-ins belong to the same plug-in type and are subject to the binding restrictions of that type.

When you create or modify throttling policies in the original throttling interface or console, the data changes are synchronized to the plug-in system. However, the changes you made in the plug-in system cannot be synchronized to the throttling interface or console.

Notes

In a throttling policy, you can configure API-, user-, and application-level throttling thresholds and a time unit, which can be second, minute, hour, or day. Before using throttling policies, you need to be familiar with the following content:

A throttling policy can include the following throttling thresholds:

- **API-level throttling threshold:** The maximum number of times that an API bound with the throttling policy can be called within a time unit. This time unit can be second, minute, hour, or day. For example, you can set this threshold to 5,000 times per minute.
- **Application-level throttling threshold:** The maximum number of times that each application can call an API bound with the throttling policy within a time unit. For example, you can set this threshold to 50,000 times per hour.
- **User-level throttling threshold:** The maximum number of times that each Alibaba Cloud account can call an API bound with the throttling policy within a time unit. An Alibaba Cloud account may have multiple applications. The throttling threshold of an Alibaba Cloud account is the sum of throttling thresholds of all the applications under this account. For example, you can set the user-level throttling threshold to 500,000 times per day.

These three thresholds can exist simultaneously in a throttling policy. Note: The user-level throttling threshold cannot be greater than the API-level throttling threshold, and the application-level throttling threshold cannot be greater than the user-level throttling threshold. That is, the application-level throttling threshold must be less than or equal to the user-level throttling threshold, and the user-level threshold must be less than or equal to the API-level throttling threshold.

You can also add special applications or users to a throttling policy. The basic API-level throttling threshold in the throttling policy applies to the added special applications or users. You need to set a special throttling threshold for each special application or user, and this special throttling threshold must not be greater than the API-level throttling threshold. The application- and user-level throttling thresholds in the throttling policy do not apply to the special applications or users.

For other operations and restrictions, see the “Overview” section of [Plug-ins](#) and the “API reference” section of [Developer Guide](#).

Plug-in configurations

You can configure throttling plug-ins in the JSON or YAML format as these two formats use the same schema. You can use the `yaml to json` tool to convert the configuration format of a throttling plug-in. The following table describes a plug-in configuration template in the YAML format.

```
---
unit: SECOND #The time unit. Valid values: SECOND, MINIUTE, HOUR, and DAY.
apiDefault: 1000 #The API-level throttling threshold.
userDefault: 30 # (Optional) The default user-level throttling threshold. Value 0 indicates that no user-level
throttling is implemented. The user-level threshold must not be greater than the API-level throttling threshold.
appDefault: 30 # (Optional) The application-level throttling threshold. Value 0 indicates that no application-level
throttling is implemented. The application-level throttling threshold must not be greater than the API-level
throttling threshold.
specials: # (Optional) The special throttling settings. You can set special throttling thresholds for special applications
or users in the throttling policy.
- type: "APP" # Special throttling is implemented for special applications.
  policies:
  - key: 10123123 # An application ID. You can log on to the API Gateway console, choose Consume APIs > APPs
from the left-side navigation pane, and click an application name to go to the application details page to obtain the
application ID.
    value: 10 # The special throttling threshold to be implemented for the special application. This value cannot exceed
the API-level throttling threshold in the throttling policy.
  - key: 10123123 # An application ID.
    value: 10 # The special throttling threshold to be implemented for the special application. This value cannot exceed
the API-level throttling threshold in the throttling policy.
- type: "USER" # Special throttling is implemented for special Alibaba Cloud accounts.
  policies:
  - key: 123455 # An Alibaba Cloud account ID. You can click the user icon in the upper-right corner of the page in
Alibaba Cloud Management Console to obtain your Alibaba Cloud account ID.
    value: 100 # The special throttling threshold to be implemented for the specified Alibaba Cloud account. This value
cannot exceed the API-level throttling threshold in the throttling policy.
```

IP address-based access control

IP address-based access control is one of the API security components available in API Gateway. This

feature is used to control the IP addresses or IP address segments from which APIs can be called. You can add an IP address to the whitelist or blacklist of an API to permit or reject the requests for this API from the specified IP address.

IP address-based access control is now integrated into the plug-in system. The original throttling interface and console are still in use. IP address-based access control policies and IP address-based access control plug-ins belong to the same plug-in type and are subject to the binding restrictions of that type.

When you create or modify IP address-based access control policies in the original IP address-based access control interface or console, the data changes are synchronized to the plug-in system. However, the changes you made in the plug-in system cannot be synchronized to the original IP address-based access control interface or console.

Usage instructions

You can configure a whitelist or blacklist in an IP address-based access control plug-in.

- **Whitelist:** You can configure a whitelist that consists of IP addresses or combinations of application IDs and IP addresses. API Gateway rejects requests from non-whitelisted IP addresses for the bound APIs.
 - An IP address whitelist allows only the requests from the whitelisted IP addresses.
 - An application ID and IP address whitelist allows only the requests sent by the whitelisted applications from the whitelisted IP addresses.
- **Blacklist:** You can configure IP address blacklists to reject access from the blacklisted IP addresses.

Plug-in configurations

You can configure IP address-based access control plug-ins in the JSON or YAML format as these two formats use the same schema. You can use the `yaml to json` tool to convert the configuration format of an IP address-based access control plug-in. The following table describes a plug-in configuration template in the YAML format.

```
---
type: ALLOW # The control type. Valid values: ALLOW and REFUSE. ALLOW indicates whitelist and REFUSE indicates
blacklist.
items:
- blocks: # IP address segments.
- 78.11.12.2 # You can configure IP addresses.
- 61.3.9.0/24 # You can configure CIDR blocks.
appId: 219810 # (Optional) The ID of an application. If appId is specified, the configured IP address segments take
effect only on the specified application.
- blocks: # IP address segments.
- 79.11.12.2 # You can configure IP addresses.
```

Backend Signature

What is backend signature?

A **backend signature (formerly signature key)** is a key-secret pair that you create and issue to API Gateway. This pair works similarly to the way an account and password work. Backend services verify the requests received from API Gateway based on the key-secret pair.

The original signature key feature has been integrated into the plug-in system. The original signature key interface and console are still in use. The original signature key feature and the backend signature plug-ins belong to the same plug-in type and are subject to the binding restrictions of that type.

When you create or modify keys in the original signature key interface or console, the data changes are synchronized to the plug-in system. However, the changes you made in the plug-in system cannot be synchronized to the original signature key interface or console.

Usage instructions

After you bind a key to an API, the signature information is added to all the requests for the API that API Gateway sends to your backend service. The backend service must parse the signature information through symmetric calculation to authenticate API Gateway. For more information about HTTP signature, see [Backend signature demo](#).

If you want to replace the key bound to an API, modify key and secret in the backend signature plug-in bound to the API. The new key takes effect immediately after it is bound to the API.

Plug-in configurations

You can configure backend signature plug-ins in the JSON or YAML format as these two formats use the same schema. You can use the `yaml to json` tool to convert the configuration format of a backend signature plug-in. The following table describes a plug-in configuration template in the YAML format.

```
---  
key: SampleKey  
secret: SampleSecret
```

JSON Web Token (JWT)

JSON Web Token (JWT) (RFC7519) is a simple authentication method. API Gateway can host the public JSON Web Keys (JWK) of users and use these JWKs to verify and sign the JWTs in the users' requests. This makes users' development work easier.

Now, you can use JWT (OpenID Connect) plug-ins to implement the original OpenID Connect feature. If you configure a JWT (OpenID Connect) plug-in and bind it to an API for which the OpenIdConnect feature has been configured, the JWT (OpenID Connect) plug-in takes effect in place of the original OpenID Connect feature.

Usage instructions

- To configure a JWT (OpenID Connect) plug-in, you need to generate a JWK manually or by using an online JWK generator. The following example shows a sample JWK:

```
{
  "kty": "RSA",
  "e": "AQAB",
  "kid": "O9fpdhrViq2zaaaBEWZITz",
  "use": "sig",
  "alg": "RS256",
  "n": "qSVxcknOm0uCq5vGsOmaorPDzHUubBmZZ4UXj-9do7w9X1uKFXAnqfto4TepSNuYU2bA_-tzSLAGBsR-
BqvT6w9SjxakeiyQpVmexxnDw5WZwpWenUAcYrfSPeONU-0hAQwFYgqZwJQMN8ptxkd0170PFauwACox4Hfr-
9FPGy8NCoIO4MfLXzJ3mJ7xqgIZp3NIOGXz-
GIAbCf13ii7kSSStpYqN3L_zzpvXUAos1FJ9IPXRV84tIZpFVh2lmRh0h8ImK-vi42dwID_hOIzayL1Xno2R0T-
d5AwTSdneq7g-Fwu8-sj4cCRWq3bd61Zs2QOJ8iustH0vSRMYdP5oYQ"
}
```

*The preceding JWK is in the JSON format. To configure a JWT (OpenID Connect) plug-in in the YAML format, you must use a JWK in the YAML format**

- For a JWT (OpenID Connect) plug-in, you only need to configure a public key. Keep your private key confidential. Only the RSA256 algorithm is currently supported.
- If you want to configure multiple JWKs, the kid field is required. If you want to configure only one JWK, the kid field is optional.
- You can configure multiple JWKs and use them together with the jwk field.
- JWT (OpenID Connect) plug-ins retrieve JWTs based on the parameter and parameterLocation settings.
 - For example, if parameter is set to X-Token and parameterLocation is set to header, the JWT is read from the X-Token header.
 - If the corresponding parameter configured in an API has the same name as the parameter configured in parameter, parameterLocation can be ignored to avoid errors when the API is called.

Plug-in configurations

You can configure JWT (OpenID Connect) plug-ins in the JSON or YAML format as these two formats use the same schema. You can use the yamll to json tool to convert the configuration format of a JWT (OpenID Connect) plug-in. The following table describes a plug-in configuration template in the YAML format.

```

---
parameter: X-Token #The parameter from which the JWT is read. It corresponds to an API parameter.
parameterLocation: header #The location from which the JWT is read. Valid values: query and header. This
parameter is optional when Request Mode for the bound API is set to Request Parameter Mapping. It is required
when Request Mode for the bound API is set to Request Parameter Passthrough.
claimParameters: #The conversion of claims to parameters. API Gateway maps JWT claims to backend parameters.
- claimName: aud #The name of a JWT claim, which can be public or private.
parameterName: X-Aud # The name of the backend parameter that the JWT claim is mapped to.
location: header # The location of the backend parameter that the JWT claim is mapped to. Valid values: query,
header, path, and formData.
- claimName: userId #The name of a JWT claim, which can be public or private.
parameterName: userId # The name of the backend parameter that the JWT claim is mapped to.
location: query # The location of the backend parameter that the JWT claim is mapped to. Valid values: query,
header, path, and formData.
preventJtiReplay: false # Controls whether to enable the anti-replay check for jti (JWT ID). Default value: false.
#
# Public key in the JWK
jwk:
  kty: RSA
  e: AQAB
  use: sig
  kid: O8fpdhrViq2zaaaBEWZITz # The kid field is optional when only one JWK is configured. If the JWT contains a kid
  field, API Gateway checks whether the kid fields are consistent.
  alg: RS256
  n: qSVxcknOm0uCq5vGsOmaorPDzHUubBmZZ4UXj-9do7w9X1uKFXAnqfto4TepSNuYU2bA_-tzSLAGBsR-
  BqvT6w9SjxakeiyQpVmexxnDw5WZwpWenUAcYrfSPEoNU-0hAQwFYgqZwJQMN8ptxkd0170PFauwACox4Hfr-
  9FPgy8NCoIO4MfLXzJ3mJ7xqgIZp3NIOGXz-
  GIAbCf13ii7kSStpYqN3L_zzpvXUAos1FJ9IPXRv84tZpFvH2ImRh0h8ImK-vI42dwID_hOIzayL1Xno2R0T-
  d5AwTSdneP7g-Fwu8-sj4cCRWq3bd61Zs2QOJ8iustH0vSRMYdP5oYQ
#
# You can configure a maximum of five JWKs and use them together with the jwk field.
# When multiple JWKs are configured, the kid field is required. If the JWT does not contain a kid field, the
consistency check on the kid field fails.
jwks:
- jwk:
  kty: RSA
  e: AQAB
  use: sig
  kid: O9fpdhrViq2zaaaBEWZITz # The kid field is optional when only one JWK is configured. If the JWT contains a kid
  field, API Gateway checks whether the kid fields are consistent.
  alg: RS256
  n: qSVxcknOm0uCq5vGsOmaorPDzHUubBmZZ4UXj-9do7w9X1uKFXAnqfto4TepSNuYU2bA_-tzSLAGBsR-
  BqvT6w9SjxakeiyQpVmexxnDw5WZwpWenUAcYrfSPEoNU-0hAQwFYgqZwJQMN8ptxkd0170PFauwACox4Hfr-
  9FPgy8NCoIO4MfLXzJ3mJ7xqgIZp3NIOGXz-
  GIAbCf13ii7kSStpYqN3L_zzpvXUAos1FJ9IPXRv84tZpFvH2ImRh0h8ImK-vI42dwID_hOIzayL1Xno2R0T-
  d5AwTSdneP7g-Fwu8-sj4cCRWq3bd61Zs2QOJ8iustH0vSRMYdP5oYQ
- jwk:
  kty: RSA
  e: AQAB

```

```

use: sig
kid: 10fpdhrViq2zaaaBEWZITz # The kid field is optional when only one JWK is configured. If the JWT contains a kid
field, API Gateway checks whether the kid fields are consistent.
alg: RS256
n: qSVxcknOm0uCq5vGsOmaorPDzHUubBmZZ4UXj-9do7w9X1uKFXAnqfto4TepSNuYU2bA_-tzSLAGBsR-
BqvT6w9SjxakeiyQpVmexxnDw5WZwpWenUAcYrfSPeONU-0hAQwFYgqZwJQMN8ptxkd0170PFauwACox4Hfr-
9FPGy8NCoIO4MfLXzJ3mJ7xqgIZp3NIOGXz-
GIAbCf13ii7kSSStpYqN3L_zzpvXUAos1FJ9IPXRv84tIZpFVh2lmRh0h8ImK-vI42dwID_hOIzayL1Xno2R0T-
d5AwTSdneq7g-Fwu8-sj4cCRWq3bd61Zs2QOJ8iustH0vSRMYdP5oYQ

```

CORS

For more information about CORS, [click here](#).

Plug-in configurations

You can configure CORS plug-ins in the JSON or YAML format as these two formats use the same schema. You can use the `yaml to json` tool to convert the configuration format of a CORS plug-in. The following table describes a plug-in configuration template in the YAML format.

```

---
allowOrigins: api.foo.com,api2.foo.com # The allowed origins. Separate multiple origins by commas (.). Default
value: *.
AllowMethods: GET, POST, PUT#The allowed HTTP methods. Separate multiple methods by commas (.).
allowHeaders: X-Ca-RequestId # The allowed request headers. Separate multiple request headers by commas (.).
exposeHeaders: X-RC1,X-RC2 # The headers that can be exposed to XMLHttpRequest objects. Separate multiple
headers by commas (.).
allowCredentials: true # Controls whether cookies are allowed.
maxAge: 172800

```

API Gateway Caches

API Gateway locally caches backend responses of API requests. When subsequent API requests arrive, API Gateway serves the matching backend responses from the cache to the API callers without sending these requests to backend services. This greatly reduces the workload on backend services.

Usage instructions

- API Gateway caches only **GET** responses.

- API Gateway does not cache backend responses to the requests that use the default second-level domain names bound to API groups. The default second-level domain name of each API group can be accessed for up to 1,000 times each day and is used for testing purposes only.
- You can configure the following parameters in the caching plug-ins to match and serve cached responses based on different criteria:
 - **varyByApp**: Cached responses are matched and served based on the applications of API callers.
 - **varyByParameters**: Cached responses are matched and served based on the values of specified parameters. You can extract these parameters from the APIs to which the plug-ins are bound.
 - **varyByHeaders**: Cached responses are matched and served based on request headers such as Accept and Accept-Language.
- Each user is allotted a cache space of 5 MB for each region. When the cache space is full, no new responses can be cached. The cached data is automatically deleted upon expiration.
- API Gateway caches backend responses for the amount of time retrieved from the Cache-Control header in the responses. If a backend response does not contain the Cache-Control header, API Gateway uses the time specified in the Duration field of the plug-in.
- The maximum cache duration for a response is 48 hours (172,800 seconds). Cache durations longer than 48 hours are considered as 48 hours.
- API Gateway ignores the Cache-Control headers of all client requests by default. You can configure API Gateway to process the Cache-Control headers by specifying mode in clientCacheControl. The mode field can be set to any of the following values:
 - off: API Gateway ignores the Cache-Control headers of all client requests.
 - all: API Gateway processes the Cache-Control headers of all client requests.
 - app: API Gateway processes only the Cache-Control headers of client requests whose application IDs (AppId) are included in the configured apps list.
- API Gateway caches only the Content-Type headers of backend responses by default. You can useCacheableHeaders to configure API Gateway to cache other response headers.

Plug-in configurations

You can configure caching plug-ins in the JSON or YAML format as these two formats use the same schema. You can use the `yaml to json` tool to convert the configuration format of a caching plug-in. The following table describes a plug-in configuration template in the YAML format.

```

---
varyByApp: false # Controls whether to match and serve cached responses based on the application IDs of API
callers. Default value: false.
varyByParameters: # Controls whether to match and serve cached responses based on the values of specified
parameters.
- userId # The name of a backend parameter. If the backend parameter is mapped to a parameter with a different
name, set userId to the mapped parameter name.
varyByHeaders: # Controls whether to match and serve cached responses based on different request headers.
- Accept #Cached responses are matched and served based on the Accept header.

```

```

clientCacheControl: # API Gateway determines how to process the Cache-Control headers of client requests based
on the clientCacheControl settings.
mode: "app" # Valid values: off, all, and app. Default value: off. off indicates that API Gateway ignores the Cache-
Control headers of all client requests. all indicates that API Gateway processes the Cache-Control headers of all
client requests. app indicates that API Gateway processes only the Cache-Control headers of client requests whose
application IDs are included in the configured apps list.
apps: # A list of application IDs. If mode is set to app, API Gateway processes only the Cache-Control headers of
client requests whose application IDs are in this list.
- 1992323 #A sample application ID. It is not an AppKey.
- 1239922 #A sample application ID. It is not an AppKey.
cacheableHeaders: # The cacheable response headers. API Gateway caches only the Content-Type and Content-
Length headers of backend responses by default.
- X-Customer-Token # The name of a cacheable response header.
duration: 3600 # The default response cache duration. Unit: seconds.

```

routing plug-in

1. Overview

A routing plug-in can change the back-end service address, back-end request path, back-end service type, and request and system parameters of an API. Routing plug-ins can be used for multi-tenant routing and blue-green release. They can also be used to distinguish between different environments.

2. How to configure a routing plug-in

2.1 Use a template

You can use a JSON or YAML template to configure your plug-in. The two formats have the same schema. You can use a YAML to JSON converter to convert the configuration format. The following table shows a template in YAML format.

```

---
routes:
# When the AppId of an API caller is 123456, requests to the API are routed to an independent address. In this
example, the VPC access name is set as slbAddressForVip.
- name: Vip
condition: "$AppId = 123456"
backend:
type: "HTTP_VPC"
vpcAccessName: "slbAccessForVip"
# Responses that are no longer supported are returned to legacy clients. ClientVersion is a custom parameter in the
API.
- name: MockForOldClient

```

```

condition: "$ClientVersion < '2.0.5'"
backend:
type: "MOCK"
statusCode: 400
mockBody: "This version is not supported!!!"
# Blue-green release scenarios: Five percent of requests are routed to the back end of a blue-green release.
- name: BlueGreenPercent05
condition: "Random() < 0.05"
backend:
type: "HTTP"
address: "https://beta-version.api.foo.com"
constant-parameters:
- name: x-route-blue-green
location: header
value: "route-blue-green"

```

The root object of the template is `routes`, which contains multiple route objects. Each route object is used to specify a route. Each route consists of the following parts:

- **name**: the name of the route. The name must be unique within each plug-in, and can contain letters and digits. If the route is selected by the system, the back end receives a request that contains an HTTP header named `X-Ca-Routing-Name`. `X-Ca-Routing-Name` indicates the name of the route.
- **condition**: the conditional expression of the route. For more information about conditional expressions, see section 2.2. If a route request meets the conditional expression, the route is then selected by the system. A plug-in selects routes based on the order in which they are configured. The first route in the configured list that meets the conditional expression will be selected and subsequent routes will be ignored. If you configure multiple routes, ensure that they are configured in the order that meets your service expectations.
- **backend**: the description of the back end. The description is consistent with the Swagger specification files for API Gateway. For more information, see []. The back-end configurations for an API in a routing plug-in override the original back-end configurations in the API. If the back-end configurations are incomplete after the overriding, the `X-Ca-Error-Code: I504RB` error is reported to the client. When you see this error, check whether your back-end configurations are complete. For more information, see section 2.3.
- **constant-parameters**: the constant parameters of the route. The constant parameters are attached to requests that are passed to the back end through the route, and are used for back-end business logic processing. The location parameter can be set to header or query.

2.2. Conditional expressions

2.2.1 Syntax

- Conditional expressions have similar syntax to SQL expressions. The basic format is `$A = 'A'` and `'$B = 'B'`.
- Each parameter starts with `$`. You can refer to parameters defined in an API. The API mapping relationship can either be mapping or passthrough. If the API has a defined

- parameter named `query1`, the request would represent that parameter as `$query1`.
- The following constant types are supported:
 - **STRING**: the string data type. Single or double quotation marks can be used to enclose a string. For example, `"Hello"` .
 - **INTEGER**: the integer data type, such as `1001` and `-1`.
 - **NUMBER**: the floating point data type, such as `0.1` and `100.0`.
 - **BOOLEAN**: the Boolean data type. Valid values: `true` and `false`.
 - You can use `and` and `or` to connect different expressions.
 - You can use `(and)` to specify the priority of conditional expressions.
 - As a built-in function, `Random ()` can generate a **NUMBER** type parameter, which returns a random number in the range of `[0, 1)`.
 - You can use `$ CaAppId` to refer to system parameters of the current request. You can refer to system parameters without defining them in the API. However, if you have defined a parameter in the API with the same name as a system parameter, the obtained value is overwritten by the custom API parameter. The system parameters that apply to routing plugins are as follows:
 - **CaStage**: the environment to which the current API request is made. Valid values: `RELEASE`, `PRE`, and `TEST`.
 - **CaDomain**: the domain name of the current request.
 - **CaRequestHandleTime**: the time in UTC at which the current request is handled.
 - **CaAppId**: the `AppId` request parameter.
 - **CaAppKey**: the `AppKey` request parameter.
 - **CaClientIp**: the IP address of the client.
 - **CaApiName**: the name of the API.
 - **CaHttpScheme**: the protocol used for the request. Valid values: `HTTP`, `HTTPS`, and `WS`.
 - **CaClientUa**: the `UserAgent` field uploaded from the client.
 - If a non-existent parameter is used in an expression, such as `$UnknowParameter = 1`, the expression evaluates to `false`.

2.2.3 Conditional expression examples

The following expression indicates a probability of 5%:

```
Random() < 0.05
```

The current API request is made to the test environment.

```
$CaStage = 'TEST'
```

The custom parameter `UserName` is set to `Admin` and the source IP address is `47.47.74.77`.

```
$UserName = 'Admin' and $CaClientIp = '47.47.74.77'
```

The user ID of the current request can be 1001, 1098, or 2011, and the HTTPS protocol is used for the request.

```
$CaHttpScheme = 'HTTPS' and ($CaAppId = 1001 or $CaAppId = 1098 or $CaAppId = 2011)
```

2.3. Back end definition and overriding rules

The structure of the back end is consistent with the Swagger definitions imported into API Gateway. For more information, see [Create an API by importing Swagger](#). The supported back end types and configuration samples are as follows. If you do not need to change the back end type in a routing plug-in, specify only the fields you want to change.

- HTTP

```
---
backend:
  type: HTTP
  address: "http://10.10.100.2:8000"
  path: "/users/{userId}"
  method: GET
  timeout: 7000
```

- HTTP_VPC

```
---
backend:
  type: HTTP_VPC
  vpcAccessName: vpcAccess1
  path: "/users/{userId}"
  method: GET
  timeout: 10000
```

- FC

```
---
backend:
  type: FC
  fcRegion: cn-shanghai
  serviceName: fcService
  functionName: fcFunction
  arn: "acs:ram::111111111:role/aliyunapigatewayaccessingfcrole"
```

- MOCK

```
---
backend:
  type: MOCK
```

```
mockResult: "mock resul sample"
mockStatusCode: 200
mockHeaders:
- name: server
value: mock
- name: proxy
value: GW
```

2.4. Limits

- The text size of a plug-in can be a maximum of **16,384 bytes**. If this limit is exceeded, the `InvalidPluginData.TooLarge` error is reported.
- The maximum number of routing entries that can be specified in a single routing plug-in is **16**. If this limit is exceeded, the `InvalidPluginData.TooManyRoutes` error is reported.
- The maximum size of a single conditional statement is **512 bytes**. If this limit is exceeded, the `InvalidPluginData.ConditionTooLong` error is reported.
- Configuration updates in a plug-in are synchronized to all APIs bound to the plug-in in real time. The maximum update frequency of plug-in configurations is **45 seconds**. If this limit is exceeded, the `InvalidPluginData.UpdateTooBusy` error is reported.

3. Typical scenarios

3.1. Configure multi-tenant routing (different back-end service addresses are allocated based on AppId)

Users whose AppId is 10098 or 10099 are my VIP users. I want requests from these two users to be routed to an independent server cluster.

```
---
routes:
# When the AppId of an API caller is 10098 or 10099, requests to the API are routed to an independent address.
# In this example, the VPC access name is set as slbAddressForVip.
- name: Vip
condition: "$AppId = 10098 or $AppId = 10099"
backend:
type: "HTTP-VPC"
vpcAccessName: "slbAccessForVip"
```

3.2. Route API requests based on the value of the CaStage parameter (TEST, PRE, or RELEASE)

My official environment is Alibaba Cloud VPC, but I want API requests made to the test environment to be routed to my public network test server.

```
---
routes:
# API requests made to the test environment are routed to my public network server.
- name: Vip
condition: "$CaStage = 'TEST'"
backend:
type: "HTTP"
address: "https://test-env.foo.com"
```

3.3. Perform a blue-green release

I plan to direct five percent of requests to a group of Beta servers to perform a blue-green release.

```
---
routes:
# Blue-green release scenarios: Five percent of requests are routed to the back end of a blue-green release.
- name: BlueGreenPercent05
condition: "Random() < 0.05"
backend:
type: "HTTP"
address: "https://beta-version.api.foo.com"
```

Backend Signature

What Is a Signature Key

A signature key is the Key-Secret pair you create, based on which the backend service verifies the request received from the gateway. Pay attention to the following points:

1. An unchangeable region must be selected during key creation. The key can only be bound to APIs in the same region.
2. One API can be bound with only one key. The key can be replaced, modified, bound to, or unbound from the API.
3. After binding a key to an API, the signature information is added to all the requests sent from the gateway to the API at your service backend. You must resolve the signature information through symmetric calculation at the backend to verify the gateway's identity. For more information about adding signature to the HTTP service, see [Backend HTTP Service Signature](#).

Modify or Replace the Leaked Key

To modify the Key-Secret pair once a key is leaked or to substitute a key bound to an API with

another key, proceed the following steps:

1. Configure the backend to support two keys: the original key and to-be-modified or replaced key, so that the request during the switching process can pass signature verification regardless the key modification or replacement.
2. After the backend is configured, modify the key. Verify that the new Key and Secret take effect and delete the leaked or obsolete key.

Throttling

What is throttling policy

You can set throttling for APIs, users, and apps by minute, hour, or day, or you can sort out the specific users or apps with designated throttling policy. The throttling policy is described as follows:

Throttling policy contains the following dimensions:

API traffic limit	The call times within a unit time for the API bound by the policy must not exceed the set value. The time unit may be minute, hour, or day, for example, 5,000 times per minute.
App traffic limit	The call times called by each app within a unit time for an API bound to the policy must not exceed the set value, for example, 50,000 times per hour.
User traffic limit	The call times called by each Alibaba Cloud account within a unit time must not exceed the set value. An Alibaba Cloud account may have multiple apps. The traffic limit for an Alibaba Cloud account is exactly the limit on the total traffic of all apps in this account. For example, the traffic may be 500,000 times per day.

The three values can be set in one throttling policy. Note that the user traffic limit must not exceed the API traffic limit, and the app traffic limit must not exceed the user traffic limit.

In addition, you can set an additional threshold value as the traffic limit value (not

allowed to exceed the value of API traffic limit) for special apps or users. However, the basic app traffic limit and user traffic limit settings in the throttling policy are no longer applicable to the special apps or users.

An unchangeable region must be selected for the throttling policy, and the throttling policy can only be applied to APIs in the same region.

The traffic of a single IP address is restricted within 100 QPS regarding with the value of API traffic limit.

A throttling policy can be bound to multiple APIs, with the limit value and special object settings applicable to each API separately. The latest policy bound to the API overwrites the previous one and takes effect immediately.

To add a special app or user, you must obtain the app ID (AppID) or the Alibaba Mail account of the user.

On the API Gateway console, you can create, modify, delete, view, bind, and unbind a throttling policy.

Monitoring and warning

The API Gateway console provides visualized API monitoring and warning in real time. You can obtain the calling status of an API, including the calling traffic, calling method, response time, and error rate. API Gateway displays data statistics on the calling status from multiple dimensions in multiple time units, and supports query of historical data for comprehensive analysis.

You can also configure the warning method (SMS or email) and subscribe to warning information to know the API running status in real time.

Limits

Limits on API Gateway products and business.

Restrictions	Description
--------------	-------------

User restrictions on activating the API Gateway service.	To activate the service, you must complete the real-name registration.
Restrictions on the number of API groups created by a user.	Each account can have at most 100 API groups.
Restrictions on the number of APIs created by a user.	At most 1000 APIs can be created in each API group. That is, at most 100,000 (100 * 1000) APIs can be created in each account.
Restrictions on the number of independent domain names bound to an API group.	At most five independent domain names can be bound to a group.
Restrictions on the traffic for calling an API.	The traffic of a single IP address of a single user used for calling each API made available by you must not exceed 100 QPS.
The limit of the official subdomain.	When the API group is created successfully, the API gateway issues a secondary domain name for that group. You can test the API in the group by accessing the domain name, and the gateway restricts the number of visits to 1000 times per day. Please do not use the secondary domain name to provide API service directly.
Restrictions on parameter size.	The parameters of the body location (including Form and Form other forms) cannot exceed 2 Mb, and other locations (including Header and Query) cannot exceed 128 Kb.

Backend Signature Demo

Overview

API Gateway provides the backend HTTP service signature verification function. To enable backend signature, you must create a signature key and bind the key to the corresponding API. (keep this key properly. API Gateway encrypts and stores the key to guarantee the security of the key.) After backend signature is enabled, API Gateway adds signature information to the request destined to the backend HTTP service. The backend HTTP service reads the signature string of API Gateway and performs local signature calculation on the received request to check whether the gateway signature and local signature result are consistent.

All the parameters you have defined are added to the signature, including the service parameters you have entered, and constant system parameters and API Gateway system parameters (such as

CaClientIp) you have defined.

How to read the API Gateway signature

- Save the signature calculated by the gateway in the header of the request. The Header name is X-Ca-Signature.

How to add a signature at the backend HTTP service

For more information about the demo (Java) of signature calculation, see <https://github.com/aliyun/api-gateway-demo-sign-backend-java>.

The signature calculation procedure is as follows:

Organize data involved in signature adding

```
String stringToSign=  
HTTPMethod + "\n" + // All letters in the HTTPMethod must be capitalized.  
Content-MD5 + "\n" + // Check whether Content-MD5 is empty. If yes, add a linefeed "\n".  
Headers + // If Headers is empty, "\n" is not required. The specified Headers includes "\n". For more information,  
see the headers organization method described as follows.  
Url
```

Calculate the signature

```
Mac hmacSha256 = Mac.getInstance("HmacSHA256");  
byte[] keyBytes = secret.getBytes("UTF-8");  
hmacSha256.init(new SecretKeySpec(keyBytes, 0, keyBytes.length, "HmacSHA256"));  
String sign = new String(Base64.encodeBase64(Sha256.doFinal(stringToSign.getBytes("UTF-8")), "UTF-8"));
```

secret is the signature key bound to an API.

Description

Content-MD5

Content-MD5 indicates the MD5 value of the body. MD5 is calculated only when HTTPMethod is **PUT** or **POST** and the body is not a form. The calculation method is as follows:

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

Headers

Headers indicates the keys and values of the headers involved in signature calculation. Read the keys of all headers involved in signature calculation from the header of the request. The key is X-Ca-Proxy-Signature-Headers. Multiple keys are separated by commas.

Headers organization method

Rank the keys of all headers involved in signature calculation in lexicographic order, and change all uppercase letters in the key of the header to lowercase, and splice the keys in the following method:

```
String headers =
HeaderKey1.toLowerCase() + ":" + HeaderValue1 + "\n"+
HeaderKey2.toLowerCase() + ":" + HeaderValue2 + "\n"+
...
HeaderKeyN.toLowerCase() + ":" + HeaderValueN + "\n"
```

URL

URL indicates the Form parameter in the Path + Query + Body. The organization method is as follows: If Query or Form is not empty, add a ?, rank the keys of Query+Form in lexicographic order, and then splice them in the following method. If Query or Form is empty, then URL is equal to Path.

```
String url =
Path +
"?" +
Key1 + "=" + Value1 +
"&" + Key2 + "=" + Value2 +
...
"&" + KeyN + "=" + ValueN
```

Note that Query or Form may have multiple values. If multiple values exist, use the first value for signature calculation.

Debugging mode

To access and debug the backend signature conveniently, you can enable the Debug mode. The debugging procedure is as follows:

Add **X-Ca-Request-Mode = debug** to the **header** of the request destined to API Gateway.

The backend service can only read **X-Ca-Proxy-Signature-String-To-Sign** from the **header**

because the linefeed is not allowed in the HTTP Header and thereby is replaced with “|” .

NOTE: **X-Ca-Proxy-Signature-String-To-Sign** is not involved in backend signature calculation.

Verify the time stamp

When the backend verifies the time stamp of the request, the system parameter **CaRequestHandleTime** is selectable in API definition and its value is the Greenwich mean time when the gateway receives the request.

OpenID Connect authorization

OpenID Connect is a lightweight standard based on OAuth 2.0, which provides a framework for identity interaction through APIs. Compared with OAuth, OpenID Connect not only authenticates a request, but also specifies the identity of the requester.

Based on OpenID Connect, the API gateway provides two way to authenticate API request:

OpenID Connect

Comply with standard OpenID Connect, the API customer request a “Token” through “userLoginName” and “password” first. And the API gateway performs Token verification on the request when the customer call the API.

OpenID Connect & AlibabaCloudAPP

Based on OpenID Connect, the API gateway performs Appkey+Token verification on the request and authenticates the Appkey and Token. The system of the API provider issues the Token and the gateway issues the Appkey.

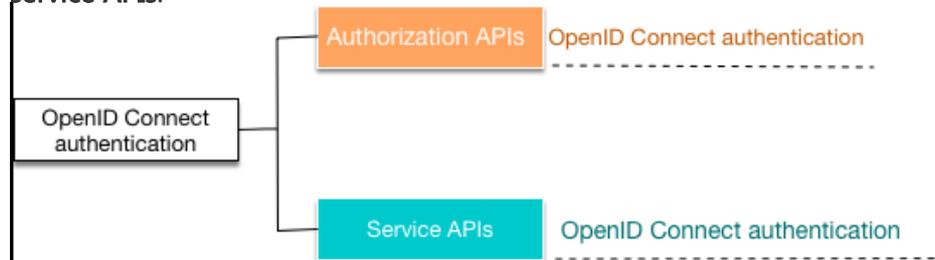
The difference between the OpenID Connect and OpenID Connect & AlibabaCloudApp: OpenID Connect & Alibaba cloud App needs to authenticate APPkey, and OpenID Connect does not.

Functions that are not supported by OpenID Connect

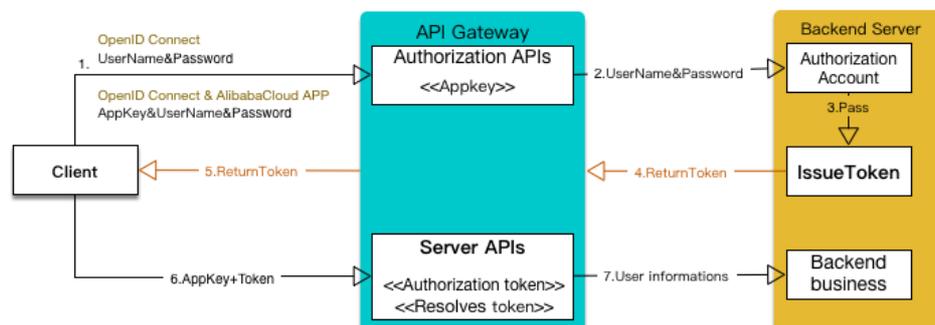
- Cannot use App authentication
- Cannot use App level Throttling
- Cannot use AlibabaCloud Account level Throttling

Implementation principle

By performing OpenID Connect authentication, APIs can be classified into **authorization APIs** and **service APIs**.



OpenID Connect authentication



- Authorization APIs: Interfaces used to issue a Token to the client. When configuring such APIs, you must inform the API gateway about the key corresponding to your Token and the public key used to resolve the Token.
- Service APIs: Interfaces used to obtain user information and perform an operation. When configuring such APIs, you must inform the API gateway about the parameter that represents the Token in your request. After the request arrives at the API gateway, the API gateway automatically checks whether this request is valid.

Certification method

The client calls an authorization API

The client uses authentications to get the "Token" :

OpenID Connect

The client uses `userLoginName/password` to call an **authorization API** to obtain authorization Token.

OpenID Connect & AlibabaCloudAPP

The client uses your Appkey signature+user name/password to call an authorization API to obtain authorization Token.

After receiving the request, the API gateway authenticates your **Appkey** first(Be effect on OpenID Connect & AlibabaCloudAPP, and OpenID Connect not). If the authentication succeeds, the API gateway calls the account system of the backend service to authenticate your **user name/password**.

After the authentication by the backend service succeeds, you can use the returned **Token** to call a **service API**.

The client calls a service API

The client uses the **Token** obtained by the **authorization API** and the **signed Appkey** to call the **service API**.

The API gateway authenticates and resolves the **Token** and sends the user information contained in the **Token** to the backend.

During this phase, the API provider must follow these steps in advance:

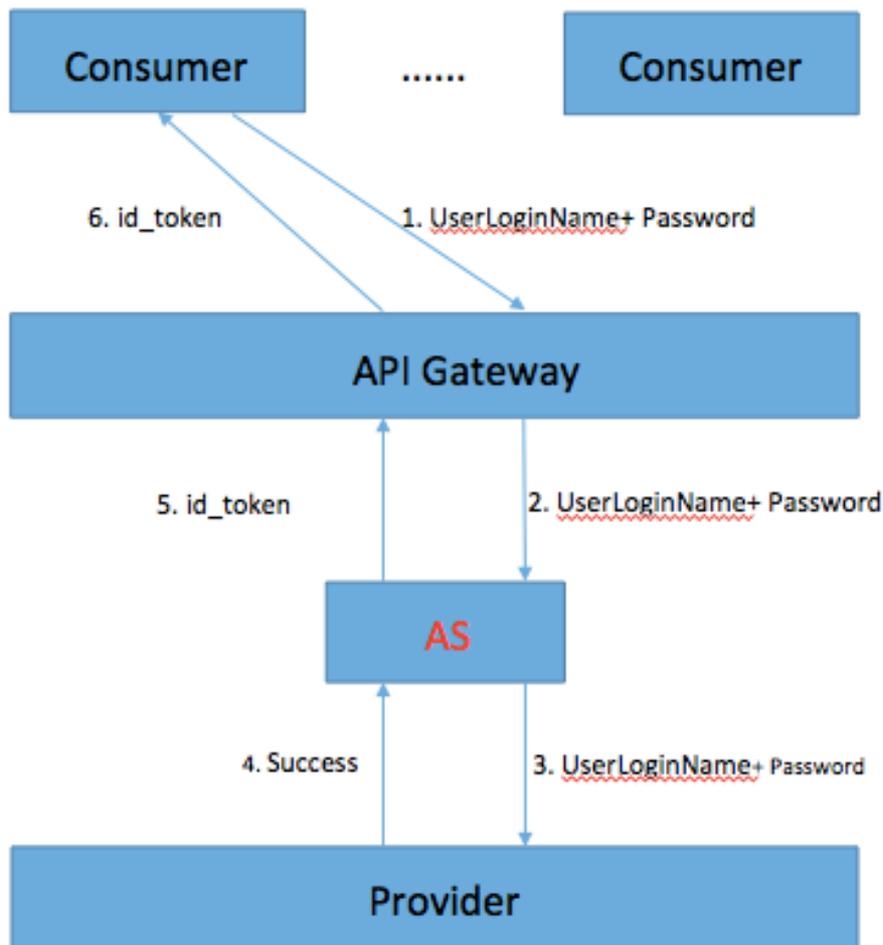
- a. Opens the account system, allows the API gateway to authenticate the **user name/password** in the request, and issues the Token based on the gateway-provided encryption mode. For more information, see **How to implement the AS module** as follows.
- b. Defines the API in the API gateway. For more information, see **Configure an API in the API gateway** as follows.
NOTE: The **user name/password** is extremely sensitive information, which is risky when being transmitted in plaintext. We recommend that you encrypt the user name/password and use the HTTPS protocol for transmission.

Solution

The solution includes two important parts:

1. Authorization server (AS): Used to generate the id_token and manage the KeyPair.

You must perform this step by yourself. For more information about the method, see **Configure an API in the API gateway** as follows.



As shown in the preceding figure, the process is as follows:

1. The Consumer (caller) sends an id_token authentication request to the API gateway, for example, in the user name+password (U+P) mode.
2. The API gateway transparently transmits the request to the AS.
3. The AS sends the user authentication request to the Provider (service provider).
4. The Provider returns the authentication results or an error message if the authentication fails.
5. If the authentication succeeds, the AS generates an id_token, which includes the User information (expandable, and can include other necessary information).

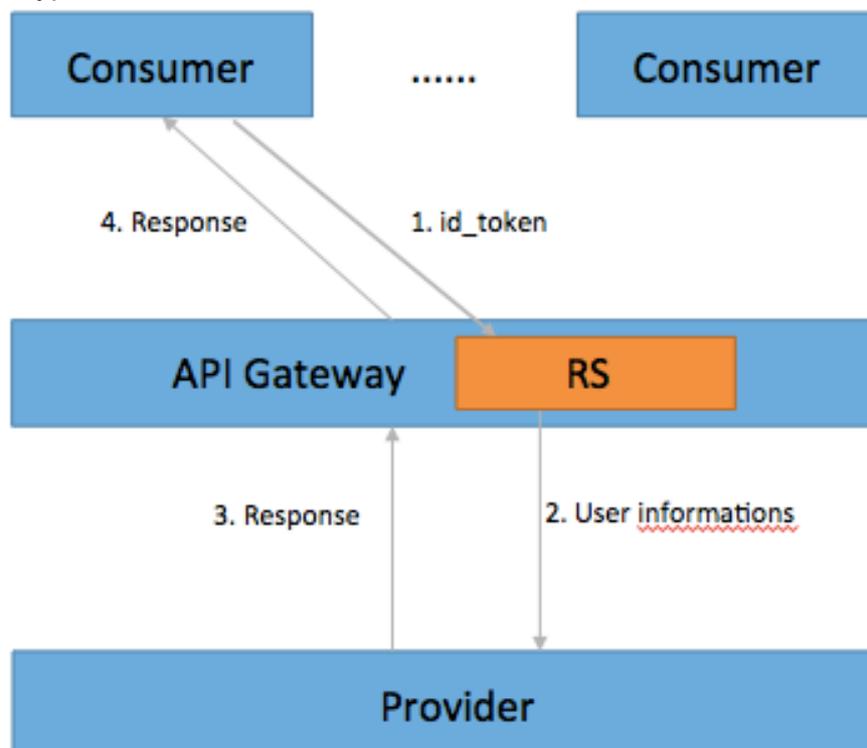
The API gateway sends the id_token returned by the AS to the Consumer.

Note: The AS is not required to be independently deployed. It can be integrated in the Provider and used to generate the id_token in the entire system. The generated id_token must meet the Specification in the OIDC protocol (version 1.0).

2. Resource server (RS): Used to verify the id_token and resolve

corresponding information.

This part is implemented by the gateway. Because the RS function has been integrated in the API gateway, the Provider only needs to generate the id_token in compliance with the corresponding encryption rules.



As shown in the preceding figure, the process is as follows:

1. The Consumer sends the parameter with the id_token to the API gateway.
2. The API gateway saves the publicKey used for verification, verifies and resolves the id_token to obtain the User information, and sends the User information to the Provider. If the authentication fails, the API gateway returns an error message.
3. The Provider processes the request and returns the results to the API gateway.
4. The API gateway transparently transmits the results from the Provider to the Consumer.

NOTE: The RS serves as the Consumer of the id_token. The request can be forwarded to the Provider only when the id_token verification succeeds.

How to implement the AS module

Use the OIDC in the AS to generate the id_token

- The id_token, also known as ID Token, is a type of tokens defined in the OIDC protocol. For

- more information, see [OpenID Connect Core 1.0](#).
- The KeyPair, keyId, and Claims are required to generate the id_token (for more information about the Claims, see [ID_Token](#)).

KeyId description

The KeyId must be unique. For example, the KeyId generated using the UUID is a string of at least 32 random characters, which can be all numbers or numbers and letters.

Example (Java)

```
String keyId = UUID.randomUUID().toString().replaceAll("-", "");
```

Or

```
String keyId = String.valueOf(UUID.randomUUID().getMostSignificantBits()) +
String.valueOf(UUID.randomUUID().getLeastSignificantBits());
```

KeyPair description

The KeyPair is a PKI system-based public and private key pair using the asymmetric algorithm. Each pair contains a publicKey and a privateKey. The publicKey is stored in the RS, which is used for verification. The privateKey is stored in the AS, which serves as the digital signature when the id_token is generated.

The KeyPair uses the RSA SHA256 encryption algorithm. To guarantee security, 2,048 bits are encrypted.

All KeyPairs used in the AS are in the JSON format. The following is an example:

publicKey:

```
{"kty":"RSA","kid":"67174182967979709913950471789226181721","alg":"ES256","n":"oH5WunqaaIopfOFBz9RfBVVII
cmk0WJDJagAcROKFiLJScQ8N\_nrexgbCMLu-dSCUWq7XMnp1ZSqw-XBS2-XEy4W4I2Q7rx3qDWY0cP8pY83hqxTZ6-
8GERJm\_0yOzR4WO4plIVVWt96-
mxn3ZgK8kmaeotkS0zS0pYMb4EEOxFFnGFqjCThuO2pimF0imxiEWw5WCdRez1v8RW72WdEflpTLJEOpP1FsFyG3OI
DbTYOqowD1YQEf5Nk2TqN\_7pYrGRKsK3BPpw4s9aXHbGrpwsCRwYbKYbmeJst8MQ4AgcorE3NPmp-
E6RxA5jLQ4axXrwC0T458LIVhypWhDqejUw","e":"AQAB"}
```

privateKey:

```
{"kty":"RSA","kid":"67174182967979709913950471789226181721","alg":"ES256","n":"oH5WunqaaIopfOFBz9RfBVVII
cmk0WJDJagAcROKFiLJScQ8N\_nrexgbCMLu-dSCUWq7XMnp1ZSqw-XBS2-XEy4W4I2Q7rx3qDWY0cP8pY83hqxTZ6-
8GERJm\_0yOzR4WO4plIVVWt96-
mxn3ZgK8kmaeotkS0zS0pYMb4EEOxFFnGFqjCThuO2pimF0imxiEWw5WCdRez1v8RW72WdEflpTLJEOpP1FsFyG3OI
DbTYOqowD1YQEf5Nk2TqN\_7pYrGRKsK3BPpw4s9aXHbGrpwsCRwYbKYbmeJst8MQ4AgcorE3NPmp-
E6RxA5jLQ4axXrwC0T458LIVhypWhDqejUw","e":"AQAB","d":"aQsHnLnOK-1xxghw2KP5JTzyJZsiwt-
ENFqjJfPuzmIYSCNAV4T39chKpkch2utd7hRtSN6Zo4NTnY8EzGQQb9yvunaiEbWUkPyJ6kM3RdlkkGLvVtp0sRwPCZ2
```

```
EAYBlSMad9jkyrtmdC0rtf9jerzt3LMLC7XWbnpC3WAl8rsRDR1CGs\_-
u4sfZfttsaUbJDD9hD0q4NfLDCVOZoQ\_8wkZxyWDAQGCe6GcCbu6N81fTp2CSVbiBj7DST\_4x2NYUA2KG8vyZYcwvi
NTxQzk4iPfdN2YQz\_9aMTZmmhVUGlmTvAjE5ebBqcqKASONfhOQHg2uR46eBKBy\_OyVOLohsQ","p":"8Tdo3DCs-
0t9JMtM0lYqPRP4wYJs37Rv6S-ygRui2MI\_hadTY9I2A199JMYw7Fjke\_wa3gqJLa98pbybdLWkrOxXbKEkwE4uc4-
fuNjLbUTC5tqdM5-
nXmpL887uREVYnk8FUzVWeXYTCNCb7OLw5I8yPJ1tR8aNcd0fJNDKkh98","q":"qlRrGSTsZzBkDgDi1xlCoYvoM76cbmx
rCUK-
mc\_kBRHfMjIHosxFUnAbxqIBE4eAJEKVfJLQrHFvIDjQb3kM9ylmwMCu9f8u9DHrT8J7LSDlLqDaXuiM2oiKtW3bAaBP
uiR7sVMFcuB5baCebHU487YymJCBTfeCZtFdi6c4w0","dp":"gVCROKonsjiQCG-s6X4j-saAL016jJsw-
7QEYE6uiMHqR\_6iJ\_uD1V8Vuec-
RxaItyc6SBsh24oeqsNoG7Ndaw7w912UVDwVjwJKQFCJdJU0v4oniItosKcPvM8M0TDUB1qZojumCWWRYsJjNSWcvA
QA7JoBAd-h6I8AqT39tcU","dq":"BckMQjRg2zhnjZo2Gjw\_aSFJZ8iHo7CHCi98LdlD03BB9oC\_kCYEDMLGDr8d7j3h-
llQnoQGbmN\_ZeGy1I7Oy3wpG9TEWQEDEpYK0jWb7rBK79hN8l1CqyBlvLK5oi-
uYCaiHkwRQ4RACz9huyRxlOz5VvlBixZnFXrzBHVPlk","qi":"M5NCVjSegf\_KP8kQLAudXUzi\_6X8T-
owtsG\_gB9xYVGnCsBHW8gccRocOY1Xa0KMotTWJl1AskCu-
TZhOJmrdeGpvkdulwmbIcnjA\_Fgflp4IAj4TCWmtRI6982hnC3XP2e-
nf\_z2XsPNiuOactY7W042D\_cajyyX\_tBEJaGOXM"}
```

Example of generating a KeyPair (Java)

```
import java.security.PrivateKey;

import org.jose4j.json.JsonUtil;
import org.jose4j.jwk.RsaJsonWebKey;
import org.jose4j.jwk.RsaJwkGenerator;
import org.jose4j.jws.AlgorithmIdentifiers;
import org.jose4j.jws.JsonWebSignature;
import org.jose4j.jwt.JwtClaims;
import org.jose4j.jwt.NumericDate;
import org.jose4j.lang.JsonException;

String keyId = UUID.randomUUID().toString().replaceAll("-", "");
RsaJsonWebKey jwk = RsaJwkGenerator.generateJwk(2048);
jwk.setKeyId(keyId);
jwk.setAlgorithm(AlgorithmIdentifiers.ECDSA_USING_P256_CURVE_AND_SHA256);
String publicKey = jwk.toJson(RsaJsonWebKey.OutputControlLevel.PUBLIC_ONLY);
String privateKey = jwk.toJson(RsaJsonWebKey.OutputControlLevel.INCLUDE_PRIVATE);
```

Process for generating an id_token

Use the Claims attributes (aud, sub, exp, iat, and iss) defined in the OIDC protocol and the attribute values to generate the Claims (the full name is JwtClaims).

Code example (Java)

```
JwtClaims claims = new JwtClaims();
claims.setGeneratedJwtId();
claims.setIssuedAtToNow();
//expire time
```

```

NumericDate date = NumericDate.now();
date.addSeconds(120);
claims.setExpirationTime(date);
claims.setNotBeforeMinutesInThePast(1);
claims.setSubject("YOUR_SUBJECT");
claims.setAudience("YOUR_AUDIENCE");
//Add custom parameters

claims.setClaim(key, value);

```

Use the `keyId`, `Claims`, `privateKey`, and the digital signature algorithm (RSA SHA256) to generate a JSON Web Signature (JWS).

Code example (Java)

```

JsonWebSignature jws = new JsonWebSignature();
jws.setAlgorithmHeaderValue(AlgorithmIdentifiers.RSA_USING_SHA256);
jws.setKeyIdHeaderValue(keyId);
jws.setPayload(claims.toJson());
PrivateKey privateKey = new RsaJsonWebKey(JsonUtil.parseJson(privateKeyText)).getPrivateKey();
jws.setKey(privateKey);

```

Use the JWS to obtain the value of the `id_token`.

Code example (Java)

```
String idToken = jws.getCompactSerialization();
```

Example of a generated `id_token`:

```

eyJhbGciOiJSUzI1NiIsImtpZCI6Ijg4NDgzNzI3NTU2OTI5MzI2NzAzMzA5OTA0MzUxMTg1ODE1NDg5In0.e
yJ1c2VySWQiOiIzMTU0NDI1OTY4NjI3IiwidGFnTmFtZSI6ImNvbWVzVGZzdCI6ImV4cCI6MTQ4
MDU5Njg3OSwiYXVkJjoiQWxpX0FQSV9Vc2VyIiwianRpIjoiTm9DMFVVeW5xV0N0RUFEVjNoeElzImh
dCI6MTQ4MDU5MzI3OSwiYm9mIjoiNDgwNTkzIiwiaWF0IjoiZGF0eU1hcD0ne3VzZXJJZD0zMzUw
TU0NDI1OTY4NjI3fScsIHN0YXR1c0NvZGU9JzAnLCBlcnJvcnM9J1tdJ30ifQ.V3rU2VCziSt6uTgdCktYR
sIwkMEMsO_jUHNCCIW_Sp4qQ5ExjtwNt9h9mTGKFRujk2z1E0k36smWf9PbNGTZTWmSYN8rvcQqdsupc
C6LU9r8jreA1Rw1CmmeWY4HsfBfeInr1wCFrEzI6_QOtf3raKSK9AowhzEsnYRKAYuc297gmV8qIQdevAwU
75qtg8j8ii3hZpJqTX67EteNCHZfhXn8wJckl5sHz2xPPyMqj8CGRQ1wrZEHjUmNPw-
unrUkt6neM0UrSqjlrQ25L8PEL2TNs7nGVdl6iS7Nasbj8fsERMKcZbP2RFzOZfKJuaivD306cJIpQwxfS1u2be
w

```

Configure an API in the API gateway

In the API edition function, the **OpenID Connect** option is added to Security certification of

Basic Info. The **Alibaba Cloud App** certification method is also included, which means that only authorized apps can call this API.

Security Certification Alibaba Cloud APP
 No Certification
 OpenID Connect

Visibility

After selecting **OpenID Connect** for Security certification, set **OpenID Connect mode**. The following two options are provided.

Security Certification

[How to use OpenID Connect?](#)

OpenID Connect: Authorization API
 Business API

Visibility

- i. Authorization APIs: Used to obtain the Token, for example, obtaining the Token using U+P.
- ii. Service APIs: Used by the Provider to provide services. The Consumer calls the obtained Token as an input parameter.

The **OpenID Connect** certification method is used for the preceding two types of APIs. The following section describes how to configure these two types of APIs, respectively.

For the authorization APIs, you must configure the KeyId and publicKey, as shown in the following figure.

Security Certification

[How to use OpenID Connect?](#)

OpenID Connect:

KeyId

Public key

KeyId: A unique ID corresponding to the KeyPair, which is generated by the AS. For example:

```
88483727556929326703309904351185815489
```

publicKey: Used to verify and resolve the Token, which is generated by the AS. For example:

```
{
  "kty": "RSA",
  "kid": "88483727556929326703309904351185815489",
  "alg": "ES256",
  "n": "ie0IKvKld7Y3izHcZ
emdDsVvXg5QtWtGF7XEKlLnn66R2\_3a30DikqV409OVL7Hv0EIACgCaBLEgZeGHTcdLE1xxDTna8MMBnB
NuMVghvFERCKh8uzpxlQsfcnFd5IFdJWj1x5Tscetrow6IA3h5zYx0rF5TkZzC4DclxgDmITRam0dsHBxr3uk9
m9YYBz2mX0ehjY0px7vIo7hZH2J3gODEPorlZkk3x8GPdlaA4P9OFAO4au9-zcVQop9vLirxdwDedk2p-
F9GP6UiQC9V2LTWqkVw\_oPBf9RIh8Qdi19jA8SeCfzAxJZYIbOTK8dYAFAVEFsvXCFvdaxQefwWFw",
  "e": "AQAB"
}
```

Configurations of other parameters are the same as those for common APIs, which are not described.

No matter creating an API or modifying an API, the configured KeyId and publicKey take effect only after the API is released.

For the service APIs, you must configure the **parameter corresponding to the Token**.

Security Certification:

[How to use OpenID Connect?](#)

OpenID Connect:

Token Parameter Name:

As shown in the preceding figure, the parameter corresponding to the Token is that sent to the id_token when the Consumer calls the API. The API gateway identifies, verifies, and resolves this parameter.

In the Input parameter definition area, a corresponding parameter must be defined. Otherwise, an error message is prompted, as shown in the following

Order	Param Name	Param Location	Type	Required	Default value	Example	Description	Operation
↓ ↑	IdToken	Query	String	<input type="checkbox"/>			342432	More Remove
+ Add								

figure.

- iii. Configuring the custom system parameters: The service API enables configuration of custom system parameters on the **Define API backend server** tab. One example is shown in the following figure.

Backend Service Parameter Configuration					
Order	Backend Param Name	Backend Param Location	Frontend Param Name	Frontend param Location	Frontend Param Type
↓ ↑	IdToken	Query	IdToken	Query	String

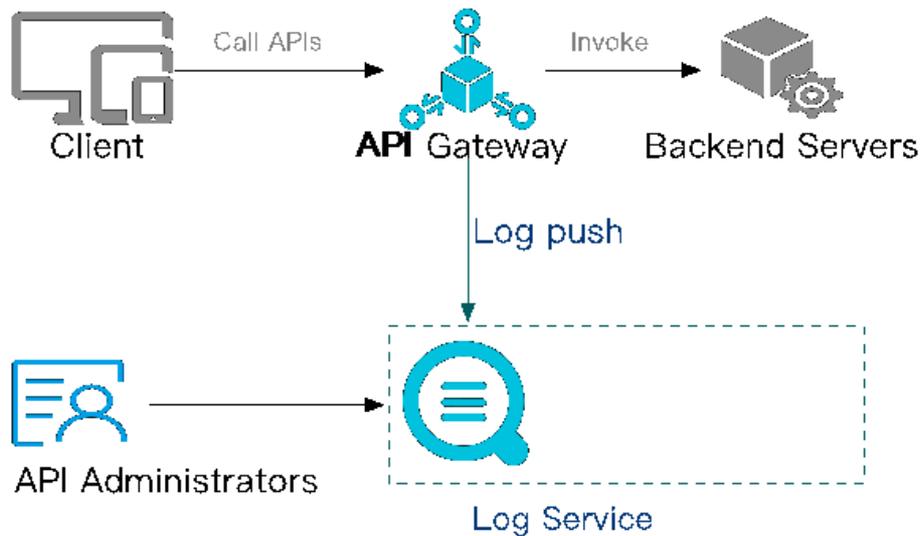
If the

`id_token` generated by the AS contains the `userId` of the Consumer, the `userId` resolved from the `id_token` sent by the Consumer is transmitted to the Provider. The configuration method for custom system parameters is similar to that for system parameters.

Besides the preceding three aspects, the method for defining other configurations of the API is the same as that in the preceding sections, which are not described.

Use Log Service to view API call logs

The API Gateway and Log Service are seamlessly integrated. The Log Service enables you to view real-time log information, download logs, and analyze logs from multiple dimensions. You can also send logs to OSS or MaxCompute.



For details about more Log Service functions, see [Log Service help](#).

You can use the Log Service free-of-charge for the first 500 MB of log data every month. For the prices of other items, see [Log Service pricing](#).

1 Function overview

1.1 Online log search

You can specify any keyword in logs to complete an exact or fuzzy log search quickly. The search results can be used for fault location or log statistics collection.

1.2 Detailed API call logs

You can obtain detailed API call information based on the following log items:

Log item	Description
apiGroupUid	API group ID
apiGroupName	API group name
apiUid	API ID
apiName	API name
apiStageUid	API environment ID
apiStageName	API environment name
httpMethod	Called HTTP method
path	Request path
domain	Called domain name
statusCode	HttpStatusCode
errorMessage	Error message
appId	Caller application ID
appName	Caller application name
clientIp	IP address of the caller client
exception	Specific error message returned from the backend
providerAliUid	API provider account ID
region	Region name, such as cn-hangzhou
requestHandleTime	Request time (UTC)
requestId	Request ID, globally unique
requestSize	Request size, unit: byte
responseSize	Returned data size, unit: byte
serviceLatency	Backend latency, unit: millisecond

1.3 Custom analysis charts

You can define statistical charts of any log items to obtain statistical data required for business operation.

1.4 Preset analysis reports

The API Gateway provides predefined statistical charts (global) for you to use directly. These statistical charts show log items including the request size, success rate, error rate, latency, number of applications that call an API, error statistics, top groups, top APIs, and top latencies.

2 Use the Log Service to view API logs

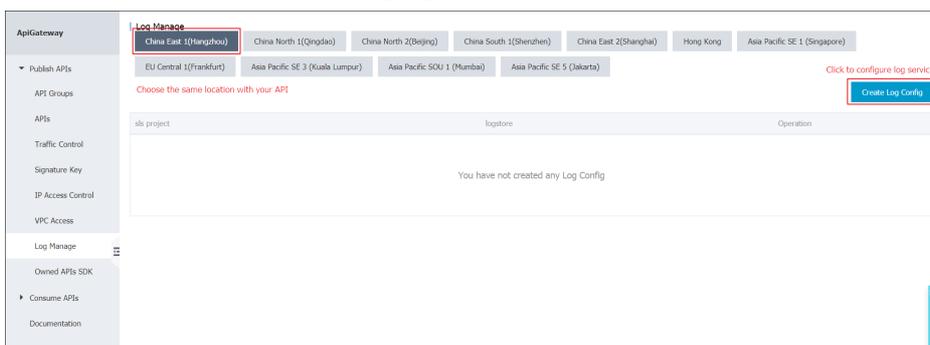
2.1 Configure the Log Service

Before using this function, make sure that you have subscribed to the log service and created a project and a logstore. Click here to create a project and logstore.

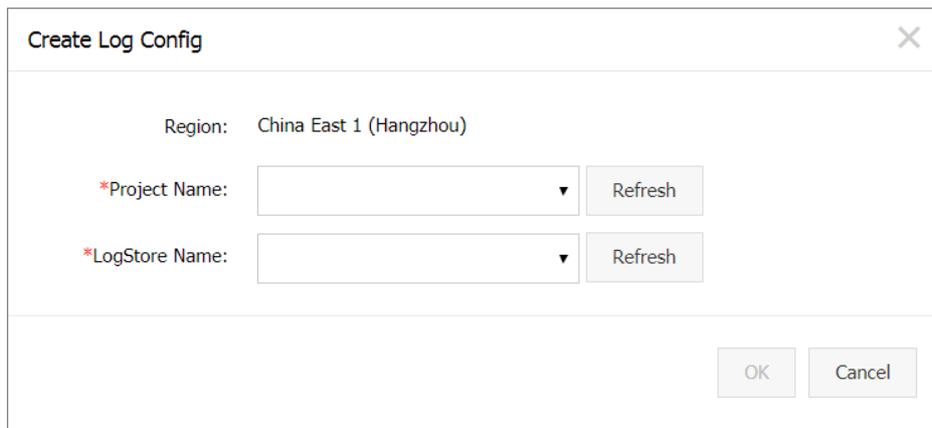
You can configure the Log Service on the API Gateway console or Log Service console.

2.1.1 Configure the Log Service on the API Gateway console

1) Open API Gateway Console and choose “Publish APIs” > “Log Manage” and select the region of your service. In the following figure, China East 1 is used as an example.

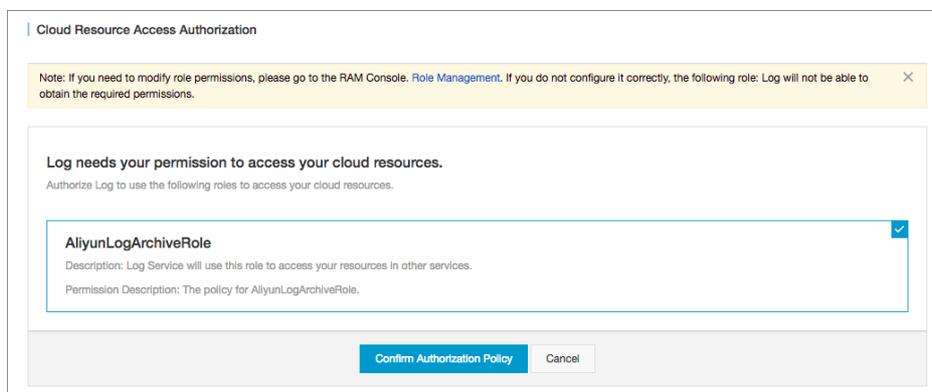


2) Click “Create Log Config” to display the log configuration page.



The image shows a 'Create Log Config' dialog box. At the top, it says 'Create Log Config' with a close button (X). Below that, the 'Region' is set to 'China East 1 (Hangzhou)'. There are two required fields: '*Project Name:' and '*LogStore Name:'. Each field is a dropdown menu with a 'Refresh' button to its right. At the bottom right, there are 'OK' and 'Cancel' buttons.

3) Select the project or logstore where the log service is required. If no options are available, click “Authorize Log Service Log Write Operation” , and then grant the authority to access your cloud resources.



The image shows a 'Cloud Resource Access Authorization' dialog box. At the top, it says 'Cloud Resource Access Authorization'. Below that, there is a yellow note: 'Note: If you need to modify role permissions, please go to the RAM Console. [Role Management](#). If you do not configure it correctly, the following role: Log will not be able to obtain the required permissions.' Below the note, there is a section titled 'Log needs your permission to access your cloud resources.' with the text 'Authorize Log to use the following roles to access your cloud resources.' Below this, there is a list of roles. The first role is 'AliyunLogArchiveRole' with a description: 'Description: Log Service will use this role to access your resources in other services.' and a permission description: 'Permission Description: The policy for AliyunLogArchiveRole.' There is a blue checkmark icon to the right of the role name. At the bottom, there are 'Confirm Authorization Policy' and 'Cancel' buttons.

4) After you confirm the authorization, the API Gateway is successfully associated with the log service.

5) Enable the indexing function to complete the configuration.

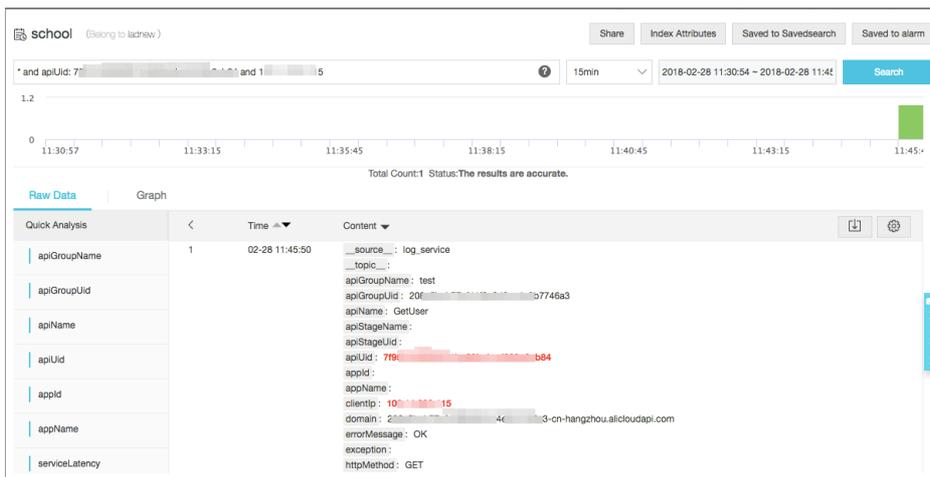
2.1.2 Configure the log service on the Log Service console

For details, see [Access logs of API Gateway](#).

After the configuration is complete, your API calls can be recorded in the logstore for the log service.

2.2 View logs

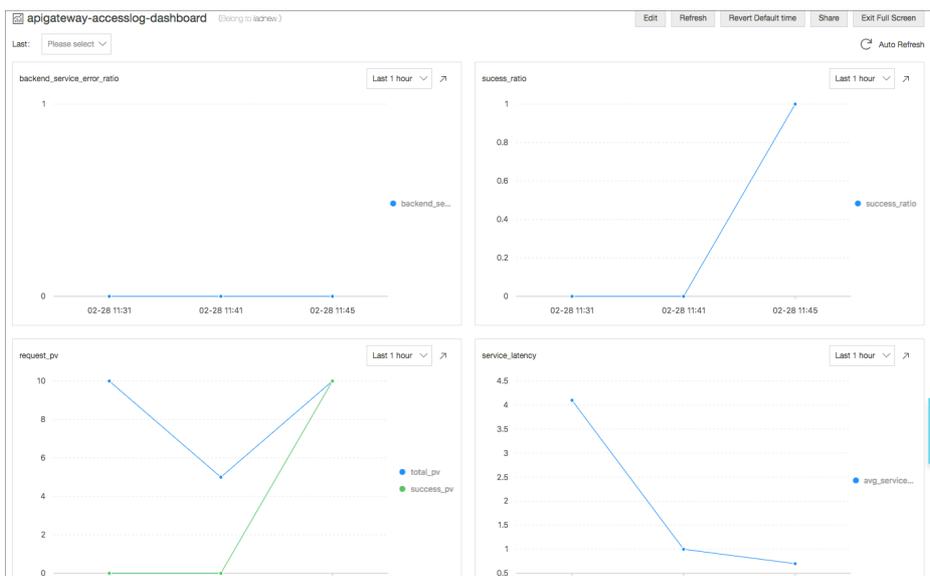
Open API Gateway console and choose “Publish APIs” > “Log Manage” > “Access Log” to go to the log console. Search for call logs online according to [Query syntax](#), as shown in the following figure.



You can also log on to the Log Service console to view logs. For details, see [Query logs](#).

2.3 View predefined reports

Predefined reports are statistical reports preset on the API Gateway to facilitate log statistics collection. Open API Gateway console and choose “Publish APIs” > “Log Manage” > “Access Log” to view the predefined reports. You can also view these predefined logs on the Log Service console.



2.4 Custom query reports

You can define query reports to meet your own business requirements. For details, see [Dashboard](#).

3 Maintain logs

Open API Gateway Console and choose “Publish APIs” > “Log Manage” to modify or delete log

configuration.

- **Modify Config:** Change the project or logstore for the log service. Then API call logs are written in the new logstore, but historical logs are still saved in the original logstore and not migrated to the new logstore.
- **Delete Config:** Delete the mapping between the API Gateway and log service. The API call logs are no longer synchronized to the log service, but the historical logs in the logstore are not deleted.

ApiGateway_RAM

The API gateway and Alibaba Cloud Resource Access Management (RAM) are integrated to enable multiple employees in an enterprise to perform permission-based API management. The API provider can create sub-accounts for employees and allow different employees to manage different APIs.

- By using the RAM, employees can use the sub-accounts to view, create, manage, and delete API groups, APIs, authorizations, and throttling policies. However, the sub-accounts are not the owner of resources, whose operation permissions may be revoked by the primary account at any time.
- Before reading this document, make sure that you have carefully read RAM help manual and API gateway API manual.
- **Skip this section if you do not have such service scenarios.**

You can use the RAM console or API to add operations.

Part 1: Policy management

The authorization policy (Policy) describes authorization content. This content contains several basic elements, including Effect, Resource, Action, and Condition.

System authorization policy

Two system permissions, AliyunApiGatewayFullAccess, and AliyunApiGatewayReadOnlyAccess, have been preset at the API gateway. You can see RAM console-policy management to check the

permissions!

Authorization Policy Name	Description	Number of References	Actions
AliyunApiGatewayFullAccess	Administrator privilege for API Gateway	1	View
AliyunApiGatewayReadOnlyAccess	Read only privilege for API Gateway	1	View

- `AliyunApiGatewayFullAccess`: It is an administrator privilege which can be used to manage all resources under the primary account, including API groups, APIs, throttling policies, and applications.
- `AliyunApiGatewayReadOnlyAccess`: It is used to view all resources under the primary account, including API groups, APIs, throttling policies, and applications, but cannot operate on them.

Custom authorization policy

You can customize management permissions precisely to an operation or resource as needed. For example, you can customize the edition permission for API `GetUsers`. You can check the defined custom authorization in RAM console-policy management-custom authorization policy. For more information about how to view, create, modify, and delete a custom authorization, see [Authorization policy management](#).

For more information about how to enter the authorization policy content, see [Policy basic elements](#), [Policy syntax structure](#), and authorization policy described as follows.

Part 2: Authorization policy

An authorization policy is a set of permissions described in the policy language. After an authorization policy is attached to a user or a group, the user or all users in the group can acquire the access permissions specified in the policy.

For more information about how to enter the authorization policy content, see [Policy basic elements](#) and [Policy syntax structure](#).

Example:

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": "apigateway:Describe*",
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

This example indicates that all the view operations are allowed.

Action (operation name list) format:

```
"Action": "<service-name>:<action-name>"
```

Among them:

- **service-name** indicates the Alibaba Cloud product name. Set this parameter to **apigateway**.
- **action-name** indicates the API name. See the following table. You can also enter the wildcards *****.

"Action": "apigateway:Describe*" indicates all the view operations.
 " Action": "apigateway:*" indicates all operations of the API gateway.

Part 3: Resource (operation object list)

A resource usually indicates an operation object, which can be API groups, throttling policies, and applications in the API gateway. The format is as follows:

```
acs:<service-name>:<region>:<account-id>:<relative-id>
```

Among them:

- **acs** is the abbreviation of Alibaba Cloud Service, indicating the Alibaba Cloud public cloud platform.
- **service-name** indicates the Alibaba Cloud product name. Set this parameter to **apigateway**.
- **region** indicates the region. You can also enter the wildcards ***** which indicate all regions.
- **account-id** indicates the account ID, such as 1234567890123456. You can also enter the wildcards *****.
- **relative-id** indicates the resource description related to the API gateway. The format is similar to a tree-like structure of a file path.

Example:

```
acs:apigateway:$regionid:$accountid:apigroup/$groupId
```

Writing:

```
acs:apigateway:*:$accountid:apigroup/
```

Check the following table by referring to API manual of the API gateway.

Action-Name	Resource
AbolishApi	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId
AddTrafficSpecialControl	acs:apigateway:\$regionid:\$accountid:trafficcontrol/\$trafficcontrolid
CreateApi	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId
CreateApiGroup	acs:apigateway:\$regionid:\$accountid:apigroup/*

CreateTrafficControl	acs:apigateway:\$regionid:\$accountid:trafficcontrol/*
DeleteAllTrafficSpecialControl	acs:apigateway:\$regionid:\$accountid:trafficcontrol/\$trafficcontrolid
DeleteApi	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupid
DeleteApiGroup	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupid
DeleteDomain	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupid
DeleteDomainCertificate	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupid
DeleteTrafficControl	acs:apigateway:\$regionid:\$accountid:trafficcontrol/\$trafficcontrolid
DeleteTrafficSpecialControl	acs:apigateway:\$regionid:\$accountid:trafficcontrol/\$trafficcontrolid
DeployApi	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupid
DescribeApi	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupid
DescribeApiError	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupid
DescribeApiGroupDetail	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupid
DescribeApiGroups	acs:apigateway:\$regionid:\$accountid:apigroup/*
DescribeApiLatency	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupid
DescribeApiQps	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupid
DescribeApiRules	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupid
DescribeApis	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupid
DescribeApisByRule	acs:apigateway:\$regionid:\$accountid:trafficcontrol/\$trafficcontrolid oracs:apigateway:\$regionid:\$accountid:secretkey/\$secretkeyid
DescribeApiTraffic	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupid
DescribeAppsByApi	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupid
AddBlackList	acs:apigateway:\$regionid:\$accountid:blacklist/

	*
DescribeBlackLists	acs:apigateway:\$regionid:\$accountid:blacklist/*
DescribeDeployedApi	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId
DescribeDeployedApis	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId
DescribeDomain	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId
DescribeDomainResolution	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId
DescribeHistoryApi	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId
DescribeHistoryApis	acs:apigateway:\$regionid:\$accountid:apigroup/*
DescribeRulesByApi	acs:apigateway:\$regionid:\$accountid:group/\$groupId
DescribeSecretKeys	acs:apigateway:\$regionid:\$accountid:secretkey/*
DescribeTrafficControls	acs:apigateway:\$regionid:\$accountid:trafficcontrol/*
ModifyApi	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId
ModifyApiGroup	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId
ModifySecretKey	acs:apigateway:\$regionid:\$accountid:secretkey/\$secretKeyId
RecoverApiFromHistorical	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId
RefreshDomain	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId
RemoveAccessPermissionByApis	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId
RemoveAccessPermissionByApps	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId
RemoveAllBlackList	acs:apigateway:\$regionid:\$accountid:blacklist/*
RemoveApiRule	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId(acs:apigateway:\$regionid:\$accountid:secretkey/\$secretKeyId oracs:apigateway:\$regionid:\$accountid:trafficcontrol/\$trafficcontrolId)
RemoveAppsFromApi	acs:apigateway:\$regionid:\$accountid:apigroup

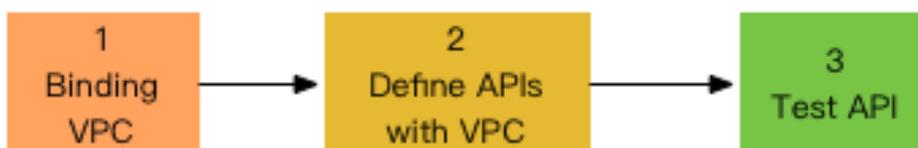
	/\$groupId
RemoveBlackList	acs:apigateway:\$regionid:\$accountid:blacklist/\$blacklistid
SetAccessPermissionByApis	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId
SetAccessPermissions	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId
SetApiRule	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId(acs:apigateway:\$regionid:\$accountid:secretkey/\$secretKeyId oracs:apigateway:\$regionid:\$accountid:trafficcontrol/\$trafficcontrolId)
SetDomain	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId
SetDomainCertificate	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId
SwitchApi	acs:apigateway:\$regionid:\$accountid:apigroup/\$groupId
CreateSecretKey	acs:apigateway:\$regionid:\$accountid:secretkey/*
DeleteSecretKey	acs:apigateway:\$regionid:\$accountid:secretkey/\$secretKeyId

Apigateway_VPC

Alibaba Cloud Virtual Private Cloud (VPC) helps you establish an isolated network environment and customize the IP address range, network segment, route table, and gateway. In addition, you can implement interconnection between VPC and traditional IDC through a leased line, VPN, or GRE to build hybrid cloud services.

The API gateway also supports open APIs for your service deployed in a VPC instance. Before reading this document, make sure that you have understood how to use VPC.

If your backend service works in a VPC instance, you must authorize the API gateway to open corresponding APIs. The process of creating an API is as follows:

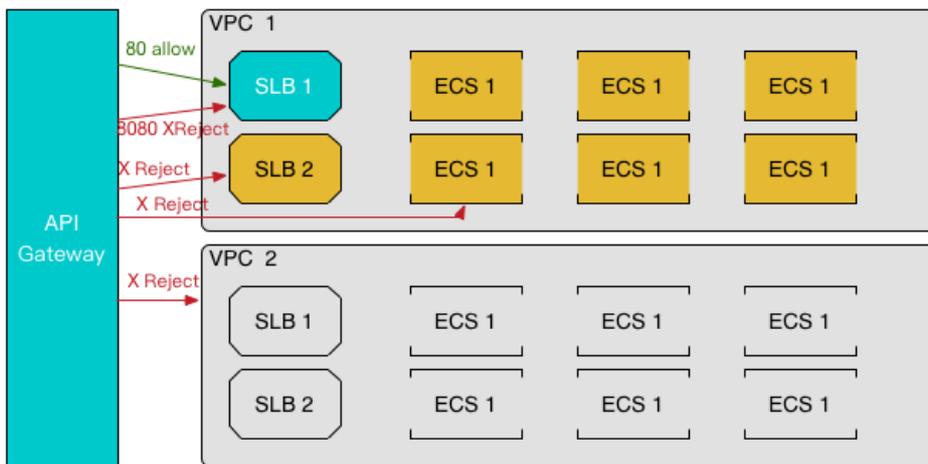


1 Authorize and bind a VPC instance

In a VPC environment, you must authorize the API gateway so that it can access the service in your VPC. During authorization, you must specify the resource and port which the API gateway can access, such as port 443 of Server Load Balancer and port 80 of ECS.

- After the authorization succeeds, the API gateway accesses resources in the VPC instance through the intranet.
- This authorization is only used for the API gateway to access corresponding backend resources.
- The API gateway cannot access unauthorized resources or ports.

For example, if only port 80 of Server Load Balancer 1 in VPC 1 is authorized to the API gateway, the API gateway can only access this port.



1.1 Prepare for a VPC environment

(1) Buy Server Load Balancer and ECS instances in the VPC environment and build the service. For more information, see [VPC user manual](#).

(2) Query the VPC information. Prepare the following VPC information:

- VPC ID: Indicates the ID of the VPC where your backend service is located.
- Instance ID: Indicates the ID of the instance of your backend service. The instance can be an ECS instance or a Server Load Balancer instance. If a Server Load Balancer instance is used, enter its instance ID.
- Port number: Indicates the number of the port that calls your backend service.

1.2 Authorize the API gateway for access

Click [API Gateway Console](#) > [Open API](#) > [Authorize VPC](#), and then click [Create Authorization](#).

Go to the authorization page and enter corresponding information.

- VPC name: Indicates the name of the authorization, which is used to select the backend address when an API is created. Make sure that this name is unique to facilitate further management.

Create VPC Access
✕

Region: China East 1 (Hangzhou)

*VPC Access Name:

It may contain Chinese characters, English letters, numbers, and English-style underlines. It must start with a letter or Chinese character and be 4-50 characters long

*VPC Id: [VPC instances](#)

It may contain English letters, numbers, and English-style underlines. It must start with a letter and be 6-20 characters long, for example: vpc-uf657qec7lx42xxxxxx

*Instance Id:

It may contain English letters, numbers, and English-style underlines. It must start with a letter and be 6-20 characters long, for example: i-uf6bzcg1pr4oxxxxxxx

*Instance Port:

It must be numbers and 2-6 characters long, for example: 80

Click **OK** to complete the authorization.

Repeat the preceding steps if you have multiple VPC instances or need to authorize multiple instances and ports.

2 Create an API

The process for creating an API is the same as that for creating other APIs. For more information, see

Create an API.

When selecting the backend service address:

- VPC channel: Set this parameter to **Use VPC channel**.
- VPC authorization: Select the created authorization as required.

Configuration of other parameters for the API is consistent with that for other APIs.

Backend Service Type HTTP/HTTPS FunctionCompute

Backend VPC Access

VPC Access [Create VPC Access](#)

Backend Request Path

The backend request path must contain the Parameter Path in the backend service parameter within brackets []. For example: /getUserInfo/[userId]

HTTP Method

Backend Timeout ms

Mock

Mock Result

Save the configuration. The API creation is complete.

3 Authorize a security group

Optional: You can skip this step if you use Server Load Balancer at the backend and have not modified the ECS security group authorization policy.

If ECS serves as the backend service of your API and you have modified the intranet inbound access policy of the security group, you must add an access policy to enable access of the following IP segments (configure the IP segments based on the region where the service is located).

Region	Direction	IP address
China East 1(Hangzhou)	Intranet inbound	100.104.13.0/24
China North 2(Beijing)	Intranet inbound	100.104.106.0/24
China South 1(Shenzhen)	Intranet inbound	100.104.8.0/24
China East 2(Shanghai)	Intranet inbound	100.104.8.0/24
Hong Kong	Intranet inbound	100.104.175.0/24
Asia Pacific SE 1 (Singapore)	Intranet inbound	100.104.175.0/24
EU Central 1(Frankfurt)	Intranet inbound	100.104.72.0/24

Asia Pacific SE 3 (Kuala Lumpur)	Intranet inbound	100.104.112.0/24
Asia Pacific SOU 1 (Mumbai)	Intranet inbound	100.104.233.0/24
Asia Pacific SE 5 (Jakarta)	Intranet inbound	100.104.72.0/24
Asia Pacific NE 1 (Tokyo)	Intranet inbound	100.104.188.0/24
Asia Pacific SE 2 (Sydney)	Intranet inbound	100.104.143.192/26

4 Test the API

You can test your API using the following methods:

- Debug the API
- Download the SDK
- Use the API to call the Demo

5 Revoke authorization

If the authorized resource or port does not provide services, delete the corresponding authorization.

5 FAQ

Is there an extra cost for using this function?

No. This function is free of charge and no extra cost is required.

Can I bind multiple VPC instances?

Yes. You can add multiple authorizations if your backend service works in multiple VPC instances.

Why cannot I authorize my VPC?

Make sure that the VPC ID, instance ID, and port number are correct and that the authorization policy and VPC are within the same region.

If I authorize the API gateway, is my VPC secure?

If you authorize the API gateway to access your VPC, the network between the gateway and VPC is connected. Security restrictions are implemented, and VPC security issues will not occur.

1. Security control authorization: Only the owner of the VPC can perform authorization.
2. Exclusive channel between the API gateway and VPC after authorization: Other persons cannot use this channel.
3. Authorization for the port of a certain resource: The gateway does not have the permission to access other ports or resources.

Configure Mock

A project is typically developed by multiple partners working together toward a specific target. The interdependence among the various stakeholders often restricts individual members during the process, and misunderstandings may adversely influence the development process or even impact the project timing. Mock can be used early in the project development cycle to simulate activities and project results. This can greatly reduce miscommunication and misunderstanding among team members in the project development and greatly improve the development efficiency.

API Gateway supports simple configuration in Mock mode.

Configure a Mock

Click [API Edition](#) > [Basic Backend Definitions](#) to configure the Mock.

Backend Service Type HTTP(s) Service VPC FunctionCompute Mock
You have used Mock, it won't invoke your backend service, please confirm it?

Backend Aone App Name
According to security department, internal user provide API must submit application security review. Otherwise, you can't publish your API. while modifying the backend uri, you should modify the backend application at the same time. If the two do not match, causing security problems, you need to take full responsibility.

Mock Configuration

Mock Result

HTTP Status Code:

Mock Header

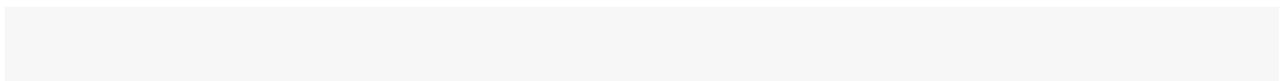
Header Name	Header Value	Operation
+ Add Item		

Backend Service Parameter Configuration

Order	Backend Param Name	Backend Param Location	Frontend Param Name	Frontend Param Location	Frontend Param Type

1. Enter the Mock response result

You can enter the actual response result in the Mock response result field. Currently, the system supports Mock response results in JSON, XML, and text formats. For example:



```
"result"
:title:"Mock test for API Gateway"
```

Save the Mock configuration and **release** it to the testing or production environment for debugging based on your needs. You can also debug on the API debugging page.

2. Enter response statusCode

The following table lists the valid values of statusCode. Format and status of HTTP 1.1 response status codes are supported. If you specify a statusCode that is not listed in the following table, the system reports an error indicating that the parameter is invalid.

http code	http message
200	OK
201	Created
202	Accepted
203	Non-Authoritative Information
204	No Content
205	Reset Content
206	Partial Content
300	Multiple Choices
301	Moved Permanently
302	Found
303	See Other
304	Not Modified
305	Use Proxy
306	(Unused)
307	Temporary Redirect
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
407	Proxy Authentication Required
408	Request Timeout

409	Conflict
410	Gone
411	Length Required
412	Precondition Failed
413	Request Entity Too Large
414	Request-URI Too Long
415	Unsupported Media Type
416	Requested Range Not Satisfiable
417	Expectation Failed
450	Parameter Required
451	Method Connect Exception
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout
505	HTTP Version Not Supported

3. Define a Mock header

API Gateway supports custom Mock headers and duplicate header names. The value of a header name cannot be empty and may only contain numbers, letters, underscores (_), and hyphens (-). The value of a header cannot be empty.

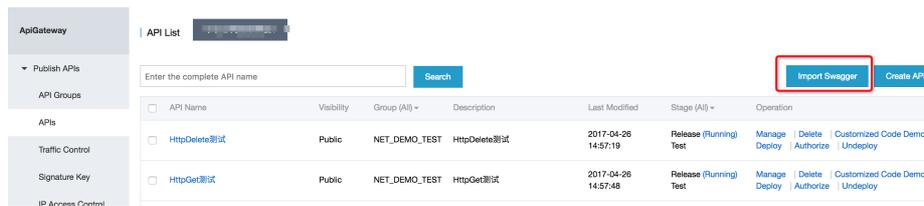
Remove a Mock

To remove a Mock, configure a different backend service. The value of the Mock response result is not removed and you can use the value for the next Mock setting. You need to **release** the change. A change takes effect only after being released.

Create APIs by Importing Swagger

Swagger is a specification used to describe API definitions, and is widely used to define and describe

APIs for backend services. You can now create APIs by importing Swagger 2.0 files into API Gateway. For more information, see [ImportSwagger](#), or operate in the console, as in the following figure:



The API Gateway Swagger extension is based on Swagger 2.0. You can create the Swagger definition for API entities, and import the Swagger file into API Gateway for bulk creations or updating API entities. By default, API Gateway supports Swagger 2.0, which is compatible with most Swagger specifications. However, these Swagger versions have some differences. For more information, see [Swagger Compatibility Reference](#).

This topic describes API Gateway extensions based on Swagger, and provides related examples to describe implementation.

Note: All parameters and valid values in Swagger are case-sensitive.

1. Swagger extensions:

The Swagger extensions are used to extend native Swagger Operation Object, providing features such as authentication, parameter mapping, and backend services. In addition, these extensions include the support for processing the ANY method in order to respond to requests made through any HTTP method. All extensions begin with `x-aliyun-apigateway-`, which are described as follows:

1.1 x-aliyun-apigateway-auth-type: authentication type

The authentication type is applied to Operation Object. The extension is used to specify the API authentication type.

Value range :

- APP (Default): the application authentication for Alibaba Cloud API Gateway.
- ANONYMOUS: Anonymous

Example:

```
...
paths:
  'path/':
    get:
      x-aliyun-apigateway-auth-type: ANONYMOUS
...
```

1.2 x-aliyun-apigateway-paramater-handling: API mapping relationship

The API mapping relationship applied to the Operation Object, is used to specify the mapping relationship between the request parameters and the backend service parameters. When you select PASSTHROUGH as the mapping relationship, the Parameter Object does not support x-aliyun-apigateway-backend-location and x-aliyun-apigateway-backend-name properties.

Value range :

- PASSTHROUGH (Default): the request parameter passthrough.
- MAPPING: the request parameter mapping.

Example:

```
...
paths:
'path/':
get:
x-aliyun-apigateway-paramater-handling: MAPPING
...
```

1.3 x-aliyun-apigateway-backend: backend type

The backend type applied to Operation Object. The parameter is used to specify backend service information. According to the backend service type, the specific properties are as follows:

1.3.1 Backend service type: HTTP

The HTTP backend type is used to configure the service address to achieve direct backend service access.

Property description:

Property name	Type	Description
type	string	Required. The value is HTTP.
address	string	Required. The address of the backend service.
path	string	Required. The path of the backend service. Support the path variable.
method	string	Required. The backend request method.
timeout	int	Optional. The default value is 10,000. The property value

		range is[500,30000]
--	--	---------------------

Example:

```

...
x-aliyun-apigateway-backend:
type: HTTP
address: http://10.10.100.2:8000
path: "/users/{userId}"
method: GET
timeout: 7,000
...

```

1.3.2 Backend service type: HTTP-VPC

The HTTP-VPC backend type is deployed when the backend service is running in the VPC network. You need to Create a VPC authorization, and then import through the VPC authorized name.

Property description:

Property name	Type	Description
type	string	Required. The value is HTTP-VPC.
vpcAccessName	string	Required. The VPC network name that is used by the backend service.
path	string	Required. Specify the backend service path. Support the path variable.
method	string	Required. The backend request method.
timeout	int	Optional. The default value is 10,000. The property value range is[500,30000]

Example:

```

...
x-aliyun-apigateway-backend:
type: HTTP_VPC
vpcAccessName: vpcAccess1
path: "/users/{userId}"
method: GET
timeout: 10,000
...

```

1.3.3 Backend service type: FC

The FC backend type is used to configure the service address to Function Compute backend service access.

Property description:

	Property name	Type	Description
	type	string	Required ; The value is FC
	fcRegion	string	Required: The region that FC belongs to
	serviceName	string	Required: The service name of current FC
	functionName	string	Required : The function name of current FC
	arn	string	Required: Arn of the current FC.

Example:

```

...
x-aliyun-apigateway-backend:
  type: FC
  fcRegion: cn-shanghai
  serviceName: fcService
  functionName: fcFunction
  arn: acs:ram::111111111:role/aliyunapigatewayaccessingfcrole
...

```

1.3.4 Backend service type: MOCK

The MOCK backend type is used to simulate the backend service call by returning the default response.

Property description:

Property name	Type	Description
type	string	Required. The value is MOCK.
mockResult	string	Required. The response result of MOCK.

Example:

```

...
x-aliyun-apigateway-backend:
type: MOCK
mockResult: mock resul sample
...

```

1.4 x-aliyun-apigateway-Constant-parameters: constant parameters

A constant parameter applied to Operation Object is used to specify the parameter applied to the backend service.

Property description:

Property name	Type	Description
backendName	string	Required. The backend parameter name.
value	string	Required. The constant value.
location	String	Required. The location of the constant parameter. You can specify[query,header]
description	string	Optional. The description of the constant parameter.

Example:

```

...
x-aliyun-apigateway-constant-parameters:
- backendName: swaggerConstant
value: swaggerConstant
location: header
description: description of swagger
...

```

1.5 x-aliyun-apigateway-system-parameters: the backend service parameters.

A backend service parameter applied to Operation Object is used to define the system parameters of the API backend service.

Property description:

Property name	Type	Description
systemName	string	Required. The system parameter name.
backendName	string	Required. The backend parameter name.
location	String	Required. The location of the constant parameter. You can specify[query,header]

Example:

```

...
x-aliyun-apigateway-system-parameters:
- systemName: CaAppId
  backendName: appId
  location: header
...

```

1.6 x-aliyun-apigateway-backend-location: the backend parameter location.

The backend parameter location is applied Parameter Object. The property applies only when the setting is x-aliyun-apigateway-paramater-handling: MAPPING. After the parameter mapping is set, the property is used to specify the parameter location when the backend service sends a request.

Value range :

- path
- header
- query
- formData

Example:

```

...
parameters:
- name: swaggerHeader
  in: header
  required: false
  type: number
  format: double
  minimum: 0.1
  maximum: 0.5
  x-aliyun-apigateway-backend-location: query
  x-aliyun-apigateway-backend-name: backendQuery
...

```

1.7 x-aliyun-apigateway-backend-name: the backend parameter name.

The backend parameter name is applied to Parameter Object. This property applies only when the setting is x-aliyun-apigateway-paramater-handling: MAPPING. After the parameter mapping is set, the property is used to specify the parameter name when the backend service sends a request.

Example:

```
...
parameters:
- name: swaggerHeader
in: header
required: false
type: number
format: double
minimum: 0.1
maximum: 0.5
x-aliyun-apigateway-backend-location: query
x-aliyun-apigateway-backend-name: backendQuery
...
```

1.8 x-aliyun-apigateway-any-method: ANY method

The ANY method is applied to Path Item Object. The method sets an API to accept any type of HTTP request.

Example:

```
...
paths:
'path/':
x-aliyun-apigateway-any-method:
...
...
```

2. Compatibility

The differences between API Gateway and the Swagger specification when defining APIs are as follows:

2.1 Comparison between Swagger parameter types and original API Gateway type

Swagger type	API Gateway type	Supported verification
--------------	------------------	------------------------

		parameters and rules
- type:integer - format:int32	Int	- minimum - maxnum
- type:integer - format:int64	Long	- minimum - maxnum
- type:number - format:float	Float	- minimum - maxnum
- type:number - format:double	Doulbe	- minimum - maxnum
- type:string	String	- maxLength - enumValues - pattern
- type:boolean - format:Boolean	Boolean	-

3. Swagger example

This topic provides three examples of Swagger extensions that are based on API Gateway. The examples cover practically all aspects of the Swagger extensions. You can refer to these examples when you define API entities based on the Swagger extensions.

Note: The examples are only for your reference.

3.1 Swagger example with HTTP as the API Gateway backend service

```
swagger: '2.0'
basePath: /
info:
  version: '0.9'
  title: Aliyun Api Gateway Swagger Sample
schemes:
  - http
  - https
paths:
  '/http/get/mapping/{userId}':
```

```
get:
operationId: case1
schemes:
- http
- https
x-aliyun-apigateway-paramater-handling: MAPPING
x-aliyun-apigateway-auth-type: ANONYMOUS
x-aliyun-apigateway-backend:
type: HTTP
address: 'http://www.aliyun.com'
path: '/builtin/echo/{userId}'
method: get
timeout: 10000
parameters:
- name: userId
in: path
required: true
type: string
- name: swaggerQuery
in: query
required: false
default: '123465'
type: integer
format: int32
minimum: 0
maximum: 100
- name: swaggerHeader
in: header
required: false
type: number
format: double
minimum: 0.1
maximum: 0.5
x-aliyun-apigateway-backend-location: query
x-aliyun-apigateway-backend-name: backendQuery
x-aliyun-apigateway-constant-parameters:
- backendName: swaggerConstant
value: swaggerConstant
location: header
description: description of swagger
x-aliyun-apigateway-system-parameters:
- systemName: CaAppId
backendName: appId
location: header
responses:
'200':
description: 200 description
'400':
description: 400 description
'/echo/test/post/{userId}':
post:
operationId: testpost
schemes:
- http
- https
x-aliyun-apigateway-paramater-handling: MAPPING
```

```
x-aliyun-apigateway-backend:  
type: HTTP  
address: 'http://www.aliyun.com'  
path: '/builtin/echo/{backend}'  
method: post  
timeout: 10000  
consumes:  
- application/x-www-form-urlencoded  
parameters:  
- name: userId  
required: true  
in: path  
type: string  
x-aliyun-apigateway-backend-name: backend  
- name: swaggerQuery1  
in: query  
required: false  
default: '123465'  
type: integer  
format: int32  
minimum: 0  
maximum: 100  
x-aliyun-apigateway-enum: 1,2,3  
- name: swaggerQuery2  
in: query  
required: false  
type: string  
x-aliyun-apigateway-backend-location: header  
x-aliyun-apigateway-backend-name: backendHeader  
- name: swaggerHeader  
in: header  
required: false  
type: number  
format: double  
minimum: 0.1  
maximum: 0.5  
x-aliyun-apigateway-backend-location: query  
x-aliyun-apigateway-backend-name: backendQuery  
- name: swaggerFormdata  
in: formData  
required: true  
type: string  
responses:  
'200':  
description: 200 description  
'400':  
description: 400 description  
x-aliyun-apigateway-any-method:  
operationId: case2  
schemes:  
- http  
- https  
x-aliyun-apigateway-paramater-handling: MAPPING  
x-aliyun-apigateway-backend:  
type: HTTP  
address: 'http://www.aliyun.com'
```

```
path: '/builtin/echo/{abc}'
method: post
timeout: 10000
parameters:
- name: userId
in: path
required: false
default: '123465'
type: integer
format: int32
minimum: 0
maximum: 100
x-aliyun-apigateway-backend-name: abc
x-aliyun-apigateway-backend-location: path
responses:
'200':
description: 200 description
'400':
description: 400 description
```

3.2 Swagger example with HTTP-VPC as the API Gateway backend service

```
swagger: '2.0'
basePath: /
info:
version: '0.9'
title: Aliyun Api Gateway Swagger Sample
schemes:
- http
- https
paths:
'/http/get/mapping/{userId}':
get:
operationId: case1
schemes:
- http
- https
x-aliyun-apigateway-paramater-handling: MAPPING
x-aliyun-apigateway-backend:
type: HTTP-VPC
vpcAccessName: vpcName1
path: '/builtin/echo/{userId}'
method: get
timeout: 10000
parameters:
- name: userId
in: path
required: true
type: string
- name: swaggerQuery
in: query
required: false
```

```
default: '123465'  
type: integer  
format: int32  
minimum: 0  
maximum: 100  
- name: swaggerHeader  
in: header  
required: false  
type: number  
format: double  
minimum: 0.1  
maximum: 0.5  
x-aliyun-apigateway-backend-location: query  
x-aliyun-apigateway-backend-name: backendQuery  
responses:  
'200':  
description: 200 description  
'400':  
description: 400 description  
'/echo/test/post':  
post:  
operationId: testpost  
schemes:  
- http  
- https  
x-aliyun-apigateway-paramater-handling: MAPPING  
x-aliyun-apigateway-backend:  
type: HTTP-VPC  
vpcAccessName: vpcName2  
path: '/builtin/echo'  
method: post  
timeout: 10000  
consumes:  
- application/x-www-form-urlencoded  
parameters:  
- name: swaggerQuery1  
in: query  
required: false  
default: '123465'  
type: integer  
format: int32  
minimum: 0  
maximum: 100  
- name: swaggerQuery2  
in: query  
required: false  
type: string  
x-aliyun-apigateway-backend-location: header  
x-aliyun-apigateway-backend-name: backendHeader  
- name: swaggerHeader  
in: header  
required: false  
type: number  
format: double  
minimum: 0.1  
maximum: 0.5
```

```

x-aliyun-apigateway-backend-location: query
x-aliyun-apigateway-backend-name: backendQuery
- name: swaggerFormdata
in: formData
required: true
type: string
responses:
'200':
description: 200 description
'400':
description: 400 description
x-aliyun-apigateway-any-method:
operationId: case2
schemes:
- http
- https
x-aliyun-apigateway-paramater-handling: PASSTHROUGH
x-aliyun-apigateway-backend:
type: HTTP-VPC
vpcAccessName: vpcName3
path: '/builtin/echo'
method: post
timeout: 10000
responses:
'200':
description: 200 description
'400':
description: 400 description

```

3.3 Swagger example with Function Compute as the API Gateway backend service

```

swagger: '2.0'
basePath: /
info:
version: '0.9'
title: Aliyun Api Gateway Swagger Sample
schemes:
- http
- https
paths:
'/http/get/mapping/{userId}':
get:
operationId: case1
schemes:
- http
- https
x-aliyun-apigateway-paramater-handling: MAPPING
x-aliyun-apigateway-backend:
type: FC
fcRegion: cn-shanghai
serviceName: fcService
functionName: fcFunction

```

```
arn: acs:ram::111111111:role/aliyunapigatewayaccessingfcrole
parameters:
- name: userId
in: path
required: true
type: string
responses:
'200':
description: 200 description
'400':
description: 400 description
```

3.4 Swagger example with MOCK as the API Gateway backend service

```
swagger: '2.0'
basePath: /
info:
version: '0.9'
title: Aliyun Api Gateway Swagger Sample
schemes:
- http
paths:
'/mock/get/mapping/{userId}':
get:
operationId: case1
schemes:
- http
- https
x-aliyun-apigateway-paramater-handling: MAPPING
x-aliyun-apigateway-backend:
type: MOCK
mockResult: mock resul sample
mockStatusCode: 200
mockHeaders:
- name: server
value: mock
- name: proxy
value: GW
parameters:
- name: userId
in: path
required: true
type: string
responses:
'200':
description: 200 description
'400':
description: 400 description
```

Environment stage management

What is environment management

Currently, all APIs are grouped into three environments: **Test**, **Pre**, and **Release**. The **Test** and **Pre** environments are used by testers to test or debug APIs. Users use APIs in the **Release** Environment.

You can add variable parameters for API groups, defining different environment stages for APIs used in the Test, Pre, and Release environments. An environment variable is a public constant that can be customized in each environment. When calling APIs, you can place environment parameters in the request. API Gateway distinguishes the request environment according to the environment parameter information in your request.

How to configure environment variable parameters

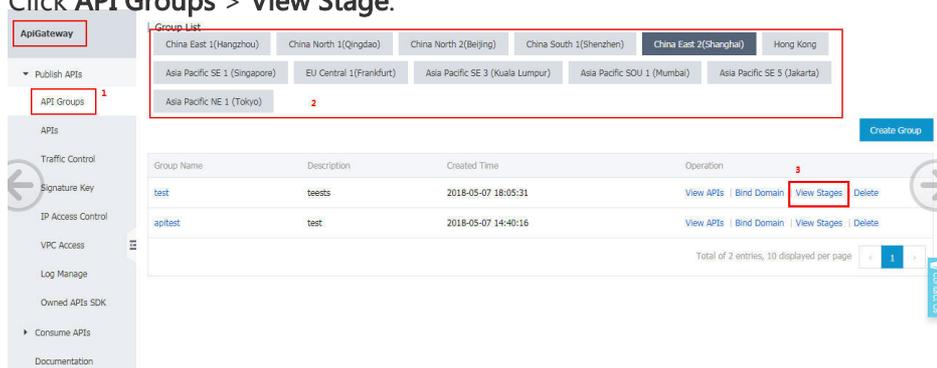
Firstly, create variables for each environment. Then, configure the created environment variables when defining APIs.

Create an environment variable

To distinguish request environments using environment variables, you must add a variable for Test, Pre, and Release environment stages respectively.

Currently, each environment allows you to configure up to 50 environment variables.

1. Log on to the API Gateway console.
2. Click **API Groups > View Stage**.



3. Select an environment stage (Release, Pre, or Test) and click **Add Variable**. You must add a variable for each environment stage one by one.

Enter the variable name and value, and click **Add**.

Name: The user-defined variable name. **Make sure that the names of the variables**

corresponding to the three environment stages are the same.

If you have multiple APIs, we recommend that the variable names indicate the actual functions to facilitate future queries.

Value: Variable value.

If Function Compute is the backend service of the API Gateway, enter the name of the service and function created in Function Compute as values of the variables. You must enter the correct names of the service and function, or else you and the other users cannot call the corresponding API. Here, we use Function Compute as an example. Assuming that we have a function service and its names in the Test, Pre, and Release environments are TestServiceD, PreServiceD, and ServiceD respectively. When you define variables for the APIs of the Test, Pre, and Release environments respectively, you can name the variable "Service" and enter the corresponding service name as the value.

You can also enter the function name as the value of the environment stage variable named as "Function" and set the respective variables for the three environments.

Configure environment variables in API definitions

When you define the APIs, add the variable in the **Request Path**, **Input Parameter Definitions**, and **Define API Backend Service**.

Expression method: #Variable Name#. For example: #Service# or #Function#. When you set Function Compute as API Gateway's backend service, you can enter the created variables as a service name

The screenshot shows the 'Basic Backend Definition' configuration page in the API Gateway console. The 'Backend Service Type' is set to 'FunctionCompute'. The 'Region' is 'China East 2 (Shanghai)'. The 'Service Name' and 'Function Name' fields are highlighted with red boxes and contain '#Service#' and '#Function#' respectively. The 'Role Arn' field is empty, and there is a 'Get Authorization' button next to it. The 'Backend Timeout' is set to 'ms' and the 'Mock' option is 'Mock Disable'.

and function name.

Call a multi-environment API

After an API is published, you call the API in different environments.

Calls to the production environment

To call Release environment APIs, you are not required to add an environment variable.

Calls to the pre-release environment

To call Pre environment APIs, add the parameter X-Ca-Stage: PRE in the header when calling the API.

Calls to the test environment

To call Test environment APIs, add the parameter X-Ca-Stage: TEST in the header when calling the API.

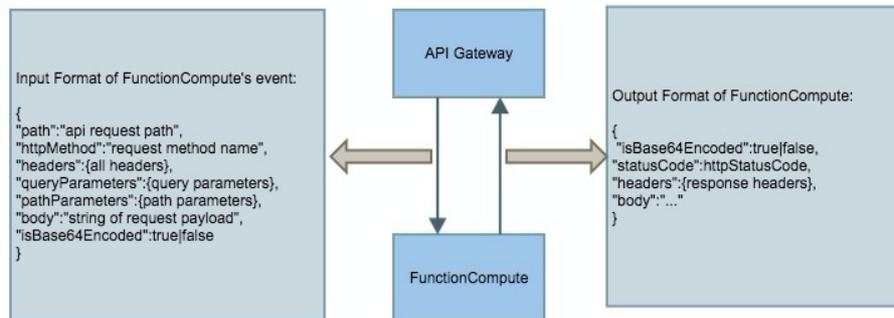
Using Function Compute as API Gateway's backend service

Function Compute is an event-driven service. Function execution can be driven by events. In other words, when a certain event occurs, it triggers a function. Currently, Function Compute supports using API Gateway as an event source. When a request sets Function Compute as the backend service API, API Gateway triggers the corresponding function and Function Compute returns the execution result to API Gateway.

API Gateway interconnects with Function Compute. This allows you to open your function services as APIs and resolves problems including certification, throttling, and data conversion (View API Gateway functions).

Implementation principles

When API Gateway calls Function Compute, the data relevant to the API is converted to Map format for transmission to Function Compute. After the data is processed by Function Compute, the statusCode, other data, body and headers are returned as output as shown in the following figure. Then, API Gateway maps the content returned by Function Compute to statusCode, header, body, and other locations to return it to the client.



Format of parameters transmitted by API Gateway to Function Compute

When Function Compute is used as a backend service of API Gateway, API Gateway uses a fixed mapping structure to send the request parameter event to Function Compute. Function Compute obtains and processes the expected parameters according to the following structure.

```
{
  "path": "api request path",
  "httpMethod": "request method name",
  "headers": {all headers, including system headers},
  "queryParameters": {query parameters},
  "pathParameters": {path parameters},
  "body": "string of request payload",
  "isBase64Encoded": "true|false, indicate if the body is Base64-encode"
}
```

Note:

- If "isBase64Encoded" is set to "true", it indicates that the API Gateway uses Base64 to encode the body content transmitted to Function Compute. Before processing the body content, Function Compute must perform Base64 decryption.
- If "isBase64Encoded" is set to "false", it indicates that the API Gateway does not use Base64 to encode the body content.

Format of parameters returned by Function Compute

Function Compute must output the content to return to API Gateway using the following JSON format, to facilitate parsing by API Gateway.

```
{
  "isBase64Encoded":true|false,
  "statusCode":httpStatusCode,
  "headers":{response headers},
  "body":"..."
}
```

Note:

- When the body content is encoded in a binary format, you must use Base64 to encode the body content in Function Compute and set "isBase64Encoded" to "true". If the body content does not need to be encoded in Base64 format, set "isBase64Encoded" to "false". When "isBase64Encoded" is "true", API Gateway performs Base64 decryption on the body content before returning it to the client.
- In a Node.js environment, Function Compute sets callback based on the specific situation.
 - To return a successful message:
callback{null,{ "statusCode" :200," body" :" ..." }}.
 - To return an exception: callback{new Error('internal server error'),null}.
 - To return a client error: callback{null,{ "statusCode" :400," body" :" param error" }}.
- If the format of the result returned by Function Compute does not conform to the format requirements, API Gateway returns 503 Service Unavailable to the client.

Create an API with Function Compute as the backend service

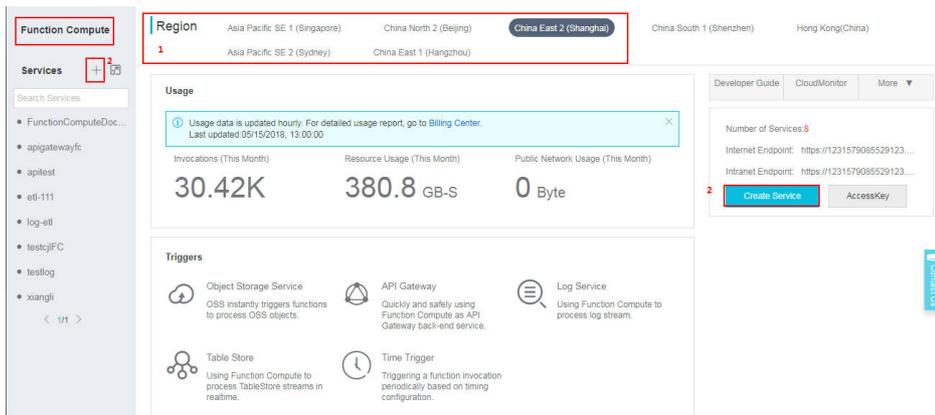
Follow these steps to create an API with Function Compute as a backend service.

1. Create a function in the Function Compute console
2. Create and define a Function Compute backend service API
3. Debug the API
4. Publish the API to the production environment

Create a function in the Function Compute console

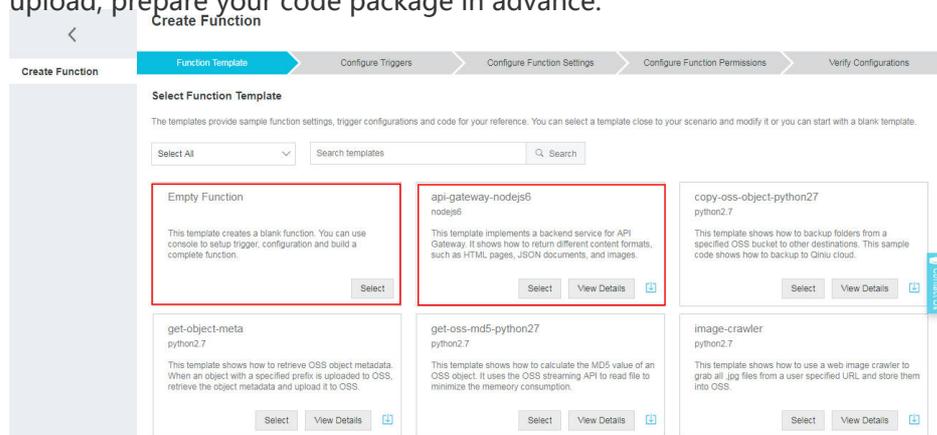
Create a service. Log on to the Function Compute console, select the **Region** of the service and function to create and click **Create Service**. In the dialog box, complete the service creation process.

Note: After creating the service, you cannot change its region, therefore, make sure you select the correct region.



2. Create a function in the created service. On the page of this newly created service, click **Create Function** to enter the function creation process:

- i. Select a function template. The Function Compute console provides an API Gateway backend implementation template for the Node.js 6 environment: `api-gateway-nodejs6`. If the `api-gateway-nodejs6` template does not suit your business needs, select **Empty Function**. After selecting the **Blank Function** template, you must provide your own code in **Basic Management Configuration**. To finish the upload, prepare your code package in advance.



- ii. Configure the trigger. Select **No Trigger** and click **Next**.
- iii. Configure function settings: Enter basic information, configure the code, set environment variables, and configure the environments. Then, click **Next**.
- iv. Ignore Service Role Management step and click **Next**. As we have already configured the corresponding role Arn permissions in the RAM console, you do not have to configure service role here. When creating an API in the API Gateway console, you click **Get Authorization** to automatically obtain the required role Arn.
- v. Check that all the information is correct and then click **Create**. After creating the function, you can review its basic information in the **Function List**.

Create and define a Function Compute backend service API

You must create an API in the API Gateway console and define its backend service as Function Compute.

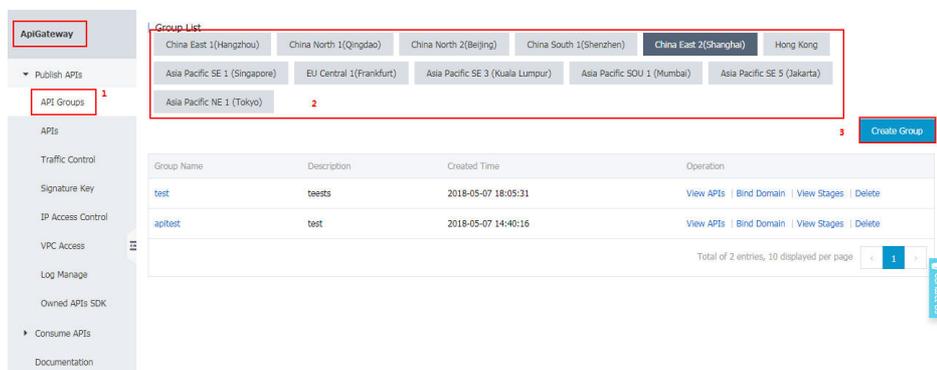
1. Log on to the API Gateway console.

Create a group.

Click **API Groups** from the left-side navigation pane, select a region for the group, and click **Create Group**. (Skip this step if you have already created a group.)

Note: If Function Compute and the API are in different regions, your Function Compute service is accessed over the Internet.

If you have high data security and network latency requirements, create the API in the same region as the function.



After creating the API group, you can use **Environment Management** to set environment variables for this group. APIs can be used in three environments: Test, Pre, and Release. To avoid backend address changes because of environment conversations, you can add environment variable parameters to implement automatic request routing. For the environment variable configuration method, see **Environment management**.

Create and define an API.

- i. After creating the group, click the **View APIs** button from the Operation column of this group to go to its **API List** page.
- ii. Click **Create API** to enter the API creation and definition process.
- iii. Enter the basic information of the API and click **Next**.

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

Name And Description

Group: test [Create Group](#)

API Name: apigateway

Security Certification: Alibaba Cloud APP

Signature Method: HmacSHA256

Visibility: Public Private
When the API group is available on the Cloud Marketplace, APIs with a type of "private" will not be available if "Private" is selected

Description: For test

[Next](#)

Define API requests and click **Next**.

Note: If the **Request Mode** is set to **Request Parameter Passthrough**, the parameter body content sent to API Gateway is not processed, and is forwarded directly to Function Compute.

ApiGateway | Basic Information | **Define API Request** | Define API Backend Service | Define Response

Basic Request Definition

Protocol: HTTP HTTPS WEBSOCKET

Custom Domain Name: [Bind domain name to the group](#)

Subdomain Name: 644f9577b4426c93470c9a96c1402-cn-hangzhou.aliyuncs.com

Request Path:
The request path must contain the Parameter Path in the request parameter within brackets ([]). For example: /getUserInfo/{userId}

HTTP Method: GET

Request Mode: Request Parameter Mapping

All request parameters must have unique names, including the dynamic parameters in the path, headers parameters, query parameters, body parameters (form parameters).

Input Parameter Definition

Order	Param Name	Param Location	Type	Required	Default value	Example	Description	Operation
+ Add								

[Prev](#) [Next](#)

Define the API backend service and click **Next**.

Note: On this page, you must:

- i. Set the **Backend Service Type** to **Function Compute**.
- ii. Enter the name of the service you created in the **Function Compute console** as the **Service Name**.
- iii. Enter the name of the function you created in the **Function Compute console** as the **Function Name**.
- iv. Click **Get Authorization** to automatically obtain the role Arn. If this is the first time you have obtained role authorization for Function

Compute as the API Gateway backend service, after you click **Get Authorization**, the RAM console's authorization page is displayed. You must click the policy to grant permission on the RAM console, and then return to the API creation page and click **Get Authorization** again. The role Arn is then automatically displayed in the selection box.

The screenshot shows the 'Define API Backend Service' step in the AWS API Gateway console. The 'Basic Backend Definition' section is active, showing the following configuration:

- Backend Service Type: **HTTP/HTTPS @ FunctionCompute**
- Backend VPC Access: **Disable**
- Backend Service Address: **Format: http(s)://host:port**
- Backend Request Path: **/use**
- HTTP Method: **GET**
- Backend Timeout: **ms**
- Mock: **Mock Disable**

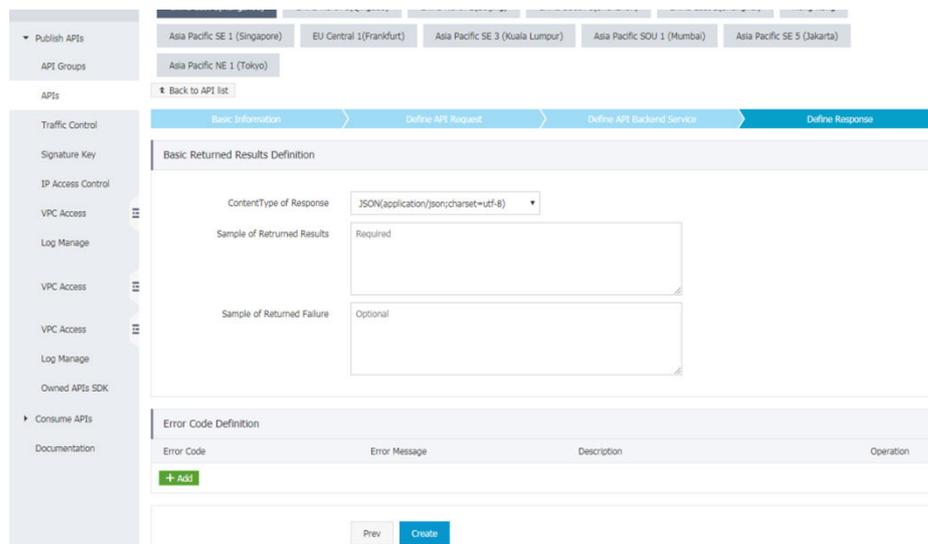
Below the basic definition, there are sections for parameter configuration:

- Backend Service Parameter Configuration**: A table with columns: Order, Backend Param Name, Backend Param Location, Frontend Param Name, Frontend param Location, Frontend Param Type.
- Constant Parameter**: A table with columns: Backend Param Name, Param Value, Param Location, Description, Operation. Includes an **Add** button.
- System Parameter**: A table with columns: System Param Name, Backend Param Name, Param Location, Description, Operation. Includes an **Add** button.

Navigation buttons **Prev** and **Next** are located at the bottom of the form.

Define response and then click **Create**.

Note: a returned result sample is required and the format must follow the **Format of parameters returned by Function Compute**.



For more information, see [Create an API](#).

Debug the API

After you create and define an API, the interface automatically displays the **API List** page. You can test whether the created API is usable and the request chain is correct.

1. On the **API List** page, click the API name or the **Manage** button to go to the **API Definition** page.
2. Click **Debug API** on the left-side navigation pane.
3. Input the request parameters and click **Send Request**. The returned results are displayed on the right-side of the page. If it returns a successful result, it indicates that the API can be used. If a 4XX or 5XX error code is returned, it indicates that the request has encountered an error. For more information, see [How to obtain the error message and Error code table](#).
4. Publish the API to the **Pre** environment for testing before it goes online. After testing proves the API is usable, you can return to the **API Definition** page and publish the API to the **Pre** environment. Then, use the subdomain name to simulate real user requests to test calling.

Note: If an environment variable is set in the API definition, enter the parameter X-Ca-Stage: RELEASE in the header to call the pre-release environment API.

Publish the API to the production environment

After you debug the API to prove it can be used, you can publish it.

1. On the **API List** page, click the API name or the **Manage** button to go to the **API Definition** page.
2. Click the **Publish** button in the upper-right corner of the page to bring up the **Publish API** dialog box.
3. Select **Release**, enter remarks, and click **Publish**. After the API is published to the production

environment, your users can call it.

For more information about publishing, see the [Publish an API document](#).

Samples

The three samples namely a function code sample, API request sample, and API Gateway return sample are described as follows.

Function code sample

This is a sample of code configured in Function Compute.

```
module.exports.handler = function(event, context, callback) {
  var responseCode = 200;
  console.log("request: " + JSON.stringify(event.toString()));
  //Converts the event to a JSON object.
  event=JSON.parse(event.toString());
  var isBase64Encoded=false;
  //Returns the result for the statusCode you enter; used to test scenarios with different statusCode values
  if (event.queryParameters !== null && event.queryParameters !== undefined) {
    if (event.queryParameters.httpStatus !== undefined && event.queryParameters.httpStatus !== null &&
    event.queryParameters.httpStatus !== "") {
      console.log("Received http status: " + event.queryParameters.httpStatus);
      responseCode = event.queryParameters.httpStatus;
    }
  }
  //If the body is Base64 encoded, Function Compute must decode the body content
  if(event.body!==null&&event.body!==undefined){
    if(event.isBase64Encoded!==null&&event.isBase64Encoded!==undefined&&event.isBase64Encoded){
      event.body=new Buffer(event.body,'base64').toString();
    }
  }
  //input is the content that API Gateway inputs to Function Compute
  var responseBody = {
    message: "Hello World!",
    input: event
  };

  //Base64 encodes the body content, can be set according to your actual needs
  var base64EncodeStr=new Buffer(JSON.stringify(responseBody)).toString('base64');

  //Format of the result that Function Compute returns to API Gateway; must conform to the following requirements:
  Set isBase64Encoded according to whether the body must be Base64 encoded
  var response = {
    isBase64Encoded:true,
    statusCode: responseCode,
    headers: {
      "x-custom-header" : "header value"
    },
    body: base64EncodeStr
  };
  console.log("response: " + JSON.stringify(response));
}
```

```
callback(null, response);
};
```

Sample request

A POST format request path for the following API:

```
/fc/test/invoke/[type]
```

```
POST http://test.alicloudapi.com/fc/test/invoke/test?param1=aaa&param2=bbb
```

```
"X-Ca-Signature-Headers":"X-Ca-Timestamp,X-Ca-Version,X-Ca-Key,X-Ca-Stage",
"X-Ca-Signature":"TnoBldxxRHrFferGlzzkGcQsaezK+ZzySloKqCOsv2U=",
"X-Ca-Stage":"RELEASE",
"X-Ca-Timestamp":"1496652763510",
"Content-Type":"application/x-www-form-urlencoded; charset=utf-8",
"X-Ca-Version":"1",
"User-Agent":"Apache-HttpClient/4.1.2 (java 1.6)",
"Host":"test.alicloudapi.com",
"X-Ca-Key":"testKey",
"Date":"Mon, 05 Jun 2017 08:52:43 GMT","Accept":"application/json",
"headerParam":"testHeader"
```

```
{"bodyParam":"testBody"}
```

API Gateway return sample

```
200
```

```
Date: Mon, 05 Jun 2017 08:52:43 GMT
```

```
Content-Type: application/json; charset=UTF-8
```

```
Content-Length: 429
```

```
Access-Control-Allow-Origin: *
```

```
Access-Control-Allow-Methods: GET,POST,PUT,DELETE,HEAD,OPTIONS , PATCH
```

```
Access-Control-Allow-Headers: X-Requested-With, X-Sequence,X-Ca-Key,X-Ca-Secret,X-Ca-Version,X-Ca-Timestamp,X-Ca-Nonce,X-Ca-API-Key,X-Ca-Stage,X-Ca-Client-DeviceId,X-Ca-Client-AppId,X-Ca-Signature,X-Ca-Signature-Headers,X-Forwarded-For,X-Ca-Date,X-Ca-Request-Mode,Authorization,Content-Type,Accept,Accept-Ranges,Cache-Control,Range,Content-MD5
```

```
Access-Control-Max-Age: 172800
```

```
X-Ca-Request-Id: 16E9D4B5-3A1C-445A-BEF1-4AD8E31434EC
```

```
x-custom-header: header value
```

```
{"message":"Hello World!","input":{"body":{"bodyParam":"testBody"},"headers":{"X-Ca-Api-Gateway":"16E9D4B5-3A1C-445A-BEF1-4AD8E31434EC","headerParam":"testHeader","X-Forwarded-For":"100.81.146.152","Content-Type":"application/x-www-form-urlencoded; charset=UTF-8"},"httpMethod":"POST","isBase64Encoded":false,"path":"/fc/test/invoke/test","pathParameters":{"type":"test"},"queryParams":{"param1":"aaa","param2":"bbb"}}
```

FAQ

Why can't I input an existing function?

You must make sure that the service and function names you enter are consistent with the names used to create the service and function on the Function Compute console.

Can I set multiple functions as an API's backend service?

No. Currently, one API can be mapped to one particular function only.

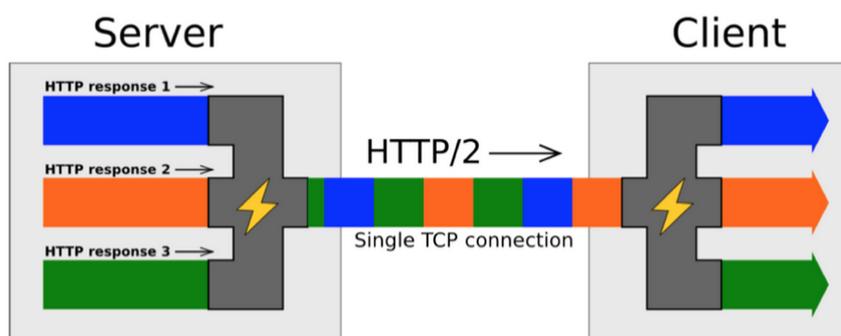
HTTP 2.0

API Gateway supports HTTP 2.0

API Gateway supports new features of HTTP 2.0, multiplexing, and request header compression.

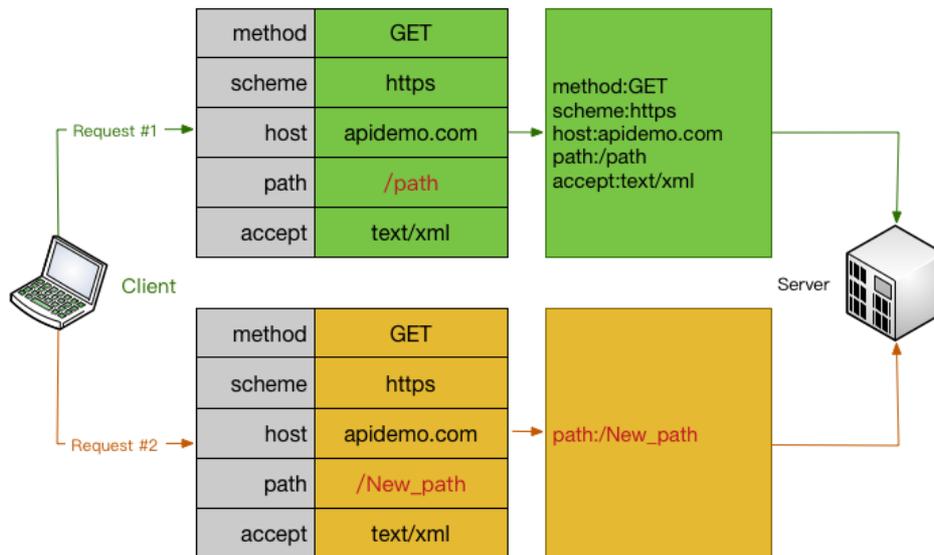
- Multiplexing: Dependency on multiple connections during concurrent processing and sending of requests and responses in HTTP 1.x is eliminated. The client and server can divide an HTTP message into multiple frames independent of each other, send the frames in a random order, and then recombine them at another end, which avoids unnecessary latency and improves efficiency. In case of a large amount of requests, the client can use this method to transmit the request data with only a few connections.

HTTP/2 Inside: multiplexing



- Header compression: As previously mentioned, the header in HTTP 1.X carries much information and must be resent each time. In HTTP 2.0, the client and server use the "header table" to trace and save the sent key-value pairs. Same data is not repeatedly sent in each request and response. The "header table" exists during the connection duration of

HTTP 2.0 and is incrementally updated by both the client and the server. Each new header key-value pair is either added to the end of the current table or replaces a value in the table, so as to reduce the data volume of each request.



How to enable HTTP 2.0

New API groups (created after July 14, 2017)

All the HTTPS APIs support HTTP2 communication between the client and API Gateway. (HTTP 2.0 runs only in an HTTPS environment, and thus you must Enable HTTPS before using HTTP 2.0.)

Stock API groups

The manual enabling function will be available in the future.

To Support HTTPS

HTTPS is a protocol integrating HTTP and SSL. It encrypts information and data to guarantee data transmission security. HTTPS is widely used today.

The API gateway also supports HTTPS to encrypt your API requests. The encryption can be API-level, that is, you can configure your APIs to support only HTTP or HTTPS or support both of them.

If you require the APIs to support HTTPS, follow these steps:

Step 1. Prepare materials

Prepare the following materials:

- A self-owned controllable domain name
- An SSL certificate applied for this domain name
- Only the PEM certificate format is supported. For more information, see [About Certificate Formats](#).

The SSL certificate contains XXXXX.key and XXXXX.pem, which can be opened using the text editor.

Example:

KEY

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEa8GjIleJ7rlo86mtbwcDnUfqzTQAm4b3zZEo1aKsfAuwcvCud
....
-----END RSA PRIVATE KEY-----
```

PEM

```
-----BEGIN CERTIFICATE-----
MIIFtDCCBJygAwIBAgIQRgWF1j00cozRI1pZ+ultKTANBgkqhkiG9w0BAQsFADBP
...
-----END CERTIFICATE-----
```

Step 2: Bind the SSL certificate

After preparing the preceding materials, log on to the API gateway console and click **Open API > Group Management**. Click the group to which the SSL certificate is to be bound and check the group details.

Before binding the SSL certificate, bind an **Independent domain name** to the API group.

The screenshot displays the 'Group Details' page for an API group named 'test_info'. The page is divided into two main sections: 'Basic Information' and 'Custom Domain Name'.

Basic Information:

- Region: China East 1 (Hangzhou)
- Group Name: test_info
- Group ID: 7
- Traffic limit (QPS): 500
- Subdomain Name: 7c...aliocloudapi.com
- Legal status: Normal
- Description: the weather test info

Custom Domain Name:

Custom Domain Name	CNAME Resolution Status	Domain Legal Status	SSL Certificate	Operation
api-...com	Unresolved	Normal	+ Add	Delete Domain
wul...com	Unresolved	Normal	fwefwef Edit	Delete Domain Delete Certificate

Independent domain name - Add an SSL certificate.

Edit Certificate
✕

*Certificate Name:

It may contain Chinese characters, English letters, numbers, and English-style underlines. It must start with a letter or Chinese character and be 4-50 characters long

*Certificate Content:

```
-----BEGIN CERTIFICATE-----
MIIC2TCCAKiCCQDCaOW7HbQyozANBgkqhkiG9w0BAQsFA
DCBqDELMakGA1UEBhMC
Q04xEDAOBgNVBAGMB0JlaUppbmcxEDAOBgNVBACMB0Jl
aUppbmcxEDAOBgNVBAMTJl

```

(pem code) [example](#)

*Private Key:

```
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQDnHUdNTZV4SeMI40AwDFJ4xVKVHlas/e
FnRCRNqasFnr1woiMc
iczShbSxt5NgvsKz7fvUAeaktKIVQ8Q72pEsUXMKsk4kbo0i

```

(pem code) [example](#)

- Certificate name: Indicates the custom name for further identification.
- Certificate content: Indicates the complete content of the certificate. You must copy all content in XXXXX.pem.
- Private key: Indicates the private key of the certificate. You must copy the content in XXXXX.key.

Click **OK** to complete binding of the SSL certificate.

Step 3: Adjust the API configuration

After binding the SSL certificate, you can enable access over HTTP, HTTPS, or HTTP and HTTPS for APIs. For security considerations, we recommend that you configure all APIs to support access over HTTPS.

Protocol

✓ HTTP

HTTPS

HTTP and HTTPS

Custom Domain Name

[REDACTED]

You can select **Open API > API list** to locate the corresponding API and click **API definition > Edit > Basic request definition** to modify the API.

The API supports the following protocols:

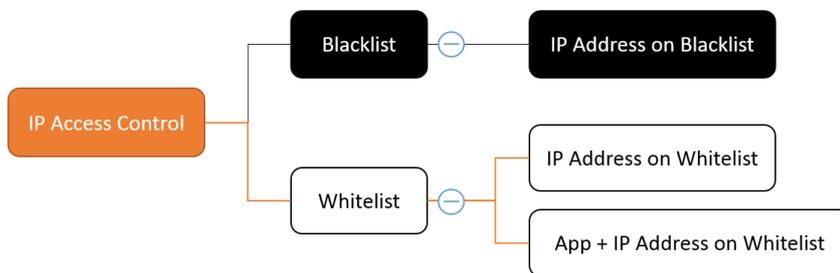
- HTTP: The API only supports access over HTTP.

- HTTPS: The API only supports access over HTTPS.
- HTTP and HTTPS: The API supports access over both HTTP and HTTPS.

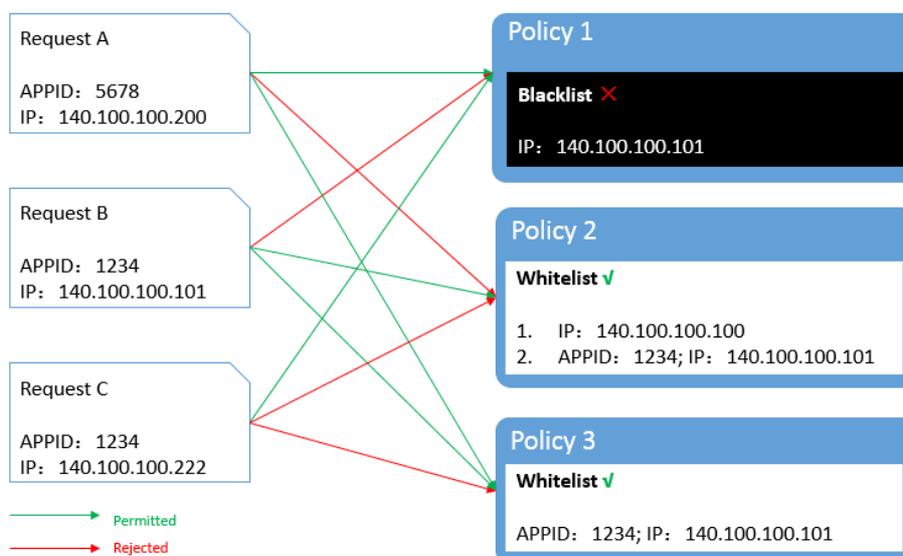
After the adjustment, the API configuration is complete. Your API supports access over HTTPS.

IP access control

IP access control is one of the API security components provided by the API Gateway and controls the source IP addresses (or IP address segments) that can call APIs. You can add an IP address to the whitelist or blacklist of an API to permit or reject the API requests from this IP address.



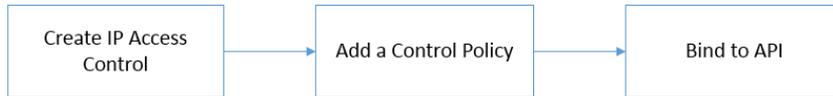
- A whitelist can contain IP addresses or its combination with application IDs. Requests from IP addresses not listed on whitelist will be rejected.
 - For IP addresses, only IP addresses from specified source are allowed to visit.
 - For IP address and application ID combinations, application IDs can only visit from their combined IP addresses. Visits from other IP addresses will be rejected.
- Requests from IP addresses on the blacklist will be rejected by API Gateway.



How to use this function

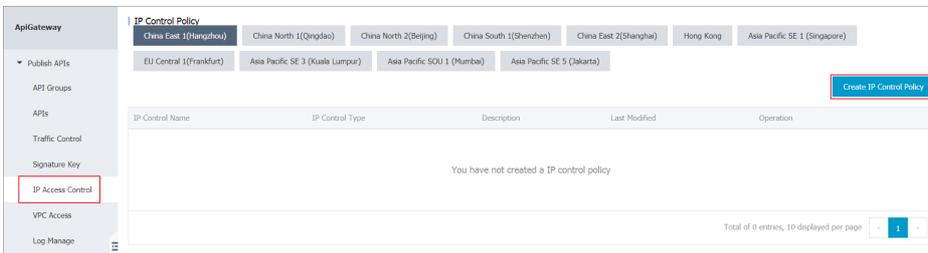
Add an IP access control policy

Create an IP access control policy and bind it to the API to which the access needs to be controlled.



Create an IP access control policy

Open API Gateway Console and choose "Publish APIs" > "IP Access Control" .



Click "Create IP Control Policy" to display the access control creation window.

The screenshot shows the 'Create IP Control' dialog box. The 'Region' is set to 'China East 1 (Hangzhou)'. The '*IP Control Name' field is empty. Below it, a note states: 'It may contain Chinese characters, English letters, numbers, and English-style underlines. It must start with a letter or Chinese character and be 4-50 characters long'. The '*IP Control Type' is set to 'Allow'. The 'Description' field is empty, with a note: 'Cannot exceed 180 characters'. At the bottom right, there are 'OK' and 'Cancel' buttons.

Enter the required information and click "OK" .

- If you set the access control type to Allow, you are configuring a whitelist.
- If you set the access control type to Refuse, you are configuring a blacklist.

Add a policy

After you create a whitelist or blacklist, you must enter the control policies corresponding to the list type. For a whitelist, you can enter the application ID, IP address, or combination of an application ID and an IP address. For a blacklist, enter an IP address.

IP Control Details [Back to IP Control List](#) Refresh

Basic Information Modify

IP Control Id: ceafe79d2b5c4df185fbf0ea7He8462	IP Control Name: Test1	Region: China East 1 (Hangzhou)
IP Control Type: Allow	Created Time: 2018-02-12 14:57:03	Modified Time: 2018-02-12 14:57:23

Description:
For: test

Policy List Bound API List Add Policy Item

Policy Item Id	AppId	Cid/Ip	Created Time	Operation
You have not add any control policy items				

Batch Delete Policies Total of 0 entries, 10 displayed per page

Add IP Control Item ×

AppId :

*IP Address:

OK Cancel

Click "OK" to complete the configuration.

API binding

Bind the IP control policy to an API for the policy to take effect.

On the IP control policy list:

IP Control Policy

China East 1(Hangzhou) China North 1(Qingdao) China North 2(Beijing) China South 1(Shenzhen) China East 2(Shanghai) Hong Kong Asia Pacific SE 1 (Singapore)

EU Central 1(Frankfurt) Asia Pacific SE 3 (Kuala Lumpur) Asia Pacific SOU 1 (Mumbai) Asia Pacific SE 5 (Jakarta) Create IP Control Policy

IP Control Name	IP Control Type	Description	Last Modified	Operation
Test1	Allow	For test	2018-02-12 14:57:23	Add Policy Item Bind API Delete IP Control

Total of 1 entries, 10 displayed per page

Find the required policy and bind API.

Bind API

Add API for the policy below:

Policy Name: Test

Select API to add:

For Test | Release | Search

API Name	Bound Policy	Operation
<input type="checkbox"/> backendRollback		<input type="button" value="+ Add"/>
<input type="checkbox"/> Test		<input type="button" value="+ Add"/>

Add selected | 2 entries in total | 1

Selected API(s) (0)

OK | Cancel

Select the corresponding API to bind the policy to it.

NOTE: Each API can have only one access control policy bound to it, no matter whether the policy is a blacklist or whitelist.

Delete an IP access control policy

Select a policy from the IP control policy list and delete it.

NOTE: If an IP control policy has been bound to an API, unbind it from the API before deleting it.

Check the bound API

You can find the API to which a policy is bound on the IP access control details page.

FAQ

When will the operation of binding or deleting an IP control policy take effect?

On the API Gateway, a policy binding operation takes effect immediately.

Can an API have different IP control policies bound in different environments?

Yes. You can bind different IP control policies to an API in different environments. We recommend that you bind a specified IP address to the test environment and pre-release environment to ensure security of the test environment.

Why is application blacklist not supported?

API calls require application authorization. To prohibit API calls for an application, you only need to delete its authorization. Therefore, application blacklist is not needed.

Invoke the API using a simple authentication method(AppCode)

1. Overview

Alibaba Cloud API Gateway authenticates client requests using a variety of methods, including Alibaba Cloud APP and OpenID Connect. Alibaba Cloud APP supports the following authentication modes:

1. Signature authentication
2. Quick authentication

For more information about signature authentication, see the “Request signature description” section of the User Guide at https://help.aliyun.com/document_detail/29475.html.

This document describes in detail how to configure and call quick authentication. Quick authentication is a much simpler process than signature authentication. It avoids the need to generate complicated signatures. Quick authentication uses the AppCode issued by API Gateway for identity authentication. The caller can put the AppCode in the request Header or in the Query parameter for quick authentication when calling the API. The process is as follows:

1. When creating an API, the API provider sets the authentication mode to Alibaba Cloud APP and select Allow AppCode authentication. All APIs in the Alibaba Cloud marketplace support AppCode by default.
2. The API caller can create applications on the Application Management page of the API Gateway console. When a user purchases an API from the marketplace, an application will be created for the user.
3. The API provider then grants API permissions to the created application. For more information about the authorization method, see https://help.aliyun.com/document_detail/29497.html.
4. The API caller logs on to the API Gateway console. On the Application Management page, they can select AppCode/AppSecret to call signature authentication or select AppCode to call quick authentication.

2. Create an API for quick authentication.

1. When creating an API, you must set Security Certification to either Alibaba Cloud APP or OpenId Connect & Alibaba Cloud APP.
2. Set AppCode Certification to Allow AppCode authentication (Header) or Allow AppCode authentication (Header & Query).

The screenshot shows the 'Basic Information' tab of the API Gateway console. The 'AppCode Certification' dropdown menu is highlighted with a blue box, showing the option 'Open after putting on cloud market'. Other visible options include 'Allow AppCode authentication (Header)' and 'Allow AppCode authentication (Header & Query)'. The 'Security Certification' is set to 'Alibaba Cloud APP' and the 'Signature Method' is 'HmacSHA256'. There are also checkboxes for 'Force Nonce Check (Anti Replay by X-Ca-Nonce)', 'Ban access to public networks Application for VPC Intranet Domain Name', and 'This API can be made public on the cloud market Cloud Market Guide'. A 'Next' button is visible at the bottom of the form.

The four AppCode

authentication options are as follows:

- Open after putting on cloud marketplace: AppCode authentication is disabled by default. After the API is available on the Alibaba Cloud marketplace, the AppCode can be put in the request Header for authentication.
- Disable AppCode authentication: AppCode authentication is disabled no matter whether the API is available on the marketplace. Users must use a signature to call this API.
- Allow AppCode authentication (Header): AppCode authentication is enabled no matter whether the API is available on the marketplace. The AppCode must be put in the request Header for authentication.
- Allow AppCode authentication (Header & Query): AppCode authentication is enabled no matter whether the API is available on the marketplace. The AppCode can be put in either the Header or the Query parameter for authentication.

Note: When defining API parameters, you do not need to define the Header or Header parameter that carries the AppCode.

3. Call the API.

After you enable AppCode authentication for the API, callers can use quick authentication to call the API without the need to implement complicated signature algorithms on their clients. This section describes how to call an API through quick authentication. An API caller can put the AppCode in either the Header or the Query parameter.

3.1 Put the AppCode in the Header.

- Add the Authorization parameter to the request Header.
- The value of the Authorization parameter is in the format of APPCODE + space + AppCode value.

Format:

```
Authorization:APPCODE AppCode value
```

Example:

```
Authorization:APPCODE 3F2504E04F8911D39A0C0305E82C3301
```

3.2 Put the AppCode in the Query parameter.

- Add the AppCode to the Query parameter of the request. The following formats of AppCode are supported: appcode, appCode, APPCODE, and APPCode.
- The value of the Authorization parameter is the AppCode value.

Example:

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

4. Risks warning

Quick authentication is simple because it does not involve the complicated signature generation processes. However, transmitting the AppCode as plaintext over the network threatens data security.

You must make sure that the client and API Gateway are using HTTPS to communicate instead of HTTP or WebSocket. Neither HTTP nor WebSocket encrypts packets for transmission, there is a large risk that the AppCode will be captured by hackers.