API Gateway

User Guide for Providers

User Guide for Providers

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.

API creation is a process to define an API request. When creating an API, you must define the format of API call requests, the format of requests sent from the gateway to backend services, the format of returned results, the parameter verification rules and so on.

Define basic information

Basic API information includes the API group, API name, description, and API type.

1. Select an API group when creating an API. An API group is a management unit of APIs with a corresponding region and domain name (for more information about the API group, see the description of groups and domain names as follows). APIs in an API group share the same region and domain name. Once selected, the group cannot be changed.
2. The API name must be unique in the group and cannot be changed.
3. Two types of APIs are required: public and private, which have no substantial difference at the public beta stage.

Define backend service information

Information of API backend services includes the type, address, and time-out time of backend services.

1. Backend service type. Only HTTP service is supported now, and Sigma, Mock, and other types of services will be supported in the future.
2. Backend service address. It is the complete IP address used by the API Gateway to call underlying services, which includes a domain name/IP+Path without Query parameter. It may contain dynamic parameters, such as username (written as username), and could be obtained only through the path entered by the caller. Therefore, do not omit these dynamic parameters when defining the final path.
3. Backend time-out time. It is the response time for beckend service to return the results after receiving requests from the gateway. The time-out time must not exceed 30 seconds.

Define the API request format

The API request format definition includes protocol and method definition, path definition, input

parameter definition, system parameter definition, parameter mapping, and parameter verification definition.

Protocol and method definition. HTTP/HTTPS protocols are supported for API calling. Methods include PUT, GET, POST, DELETE, HEAD, and MULITIPART.

Path definition. It is the path used by the caller to call the API available to external resources. The gateway stores the corresponding relations and locates the corresponding paths. The path may differ from that in the backend service address. You have to map the parameters when defining the path if they are in the backend service address.

Input parameter definition. The parameters to input conprise header, query, and body. You must define the **name** of the input parameter of the user request. Choose the **required parameters**, and provide the **example value**, **default value** and **description**. The types of parameters include **String**, **Number**, **Boolean** and **JSON**. The transmission mode of the body parameter may be **transparent transmission**.

Parameter verification definition. When defining the input parameters, you can click **More** to set verification for the parameter, including verification of the **enumeration value**, the **string length**, and **the maximum and minimum values of the number**. The gateway intercepts invalid requests, relieving burdens of your backend services.

Parameter mapping. To guise your auctual parameter name of your backend service, you can configure a backend parameter mapping for each parameter when defining the parameter.

System parameter definition. System parameters are invisible to API callers. Two types of system parameter are required, one of which that is transmitted by the gateway to you is described in the following table:

Name	Meaning
CaClientIp	The clien IP address which sends the request
CaDomain	The domain name which sends the request
CaRequestHandleTime	Request time (Greenwich mean time)
CaAppId	ID of the app which sends the request
CaRequestId	RequestId
CaApiName	API name
CaHttpSchema	The protocol (HTTP or HTTPS) used by the user to call the API
CaProxy	Proxy (AliCloudApiGateway)

When creating an API, you must configure the system parameter and select **parameter position** and **backend parameter name**.

The other type is custom system parameter required by your API backend service. It may be a constant parameter. The configuration includes the **parameter value**, **backend parameter name**, and the **parameter position** in the request.

Returned result definition. It is the type and example of returned results. Currently, the gateway does not process returned results.

Note: You must enter the **dynamic parameters** in the path, **headers parameter**, **query parameter**, **body parameter** (non-binary), **constant parameter**, and **system parameter**. The parameter name must be globally unique. It is not allowed to enter a parameter named "name" in headers and queries at the same time.

After the preceding steps, now you can test and release the API, grant permissions to your customers, bind a signature key and throttling policy to the API, and perform other security configurations.

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.

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 attentions below:

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 need to 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 will be 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.

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 unchangable region shall 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 will be added to all the requests sent from the gateway to the API at your service backend. You need to resolve the signature information through symmetric calculation at the backend to verify the gateway’ s identity. For details about adding signature to the HTTP service, refer to 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.

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 unchangable 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 appliable to each API separately. The lattest 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.

- 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 on API Gateway products and business.

Restrictions	Description
User restrictions on activating the API Gateway service.	To activate the service, you need to complete the real-name authentication.
Restrictions on the number of API groups created by a user.	Each account can have at most 50 API groups.
Restrictions on the number of APIs created by a user.	At most 200 APIs can be created in each API group. That is, at most 10,000 (50 * 200) 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 shall not exceed 100 QPS.
The limit of the official subdomain.	When the API group is created successfully, the API gateway will issue 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) can not exceed 2 Mb, and other locations (including Header and Query) can not exceed 128 Kb.

Overview

API Gateway provides the backend HTTP service signature verification function. To enable backend signature, you need to create a signature key and bind the key to the corresponding API. (keep this key properly. API Gateway encrypts and stores the key to ensure the security of the key.) After backend signature is enabled, API Gateway will add 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 will be 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 details about the demo (Java) of signature calculation, refer to 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 should 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 details, refer to the headers organization method described below.
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 there are multiple values, use the first value for signature calculation.

Debug 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 just 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.

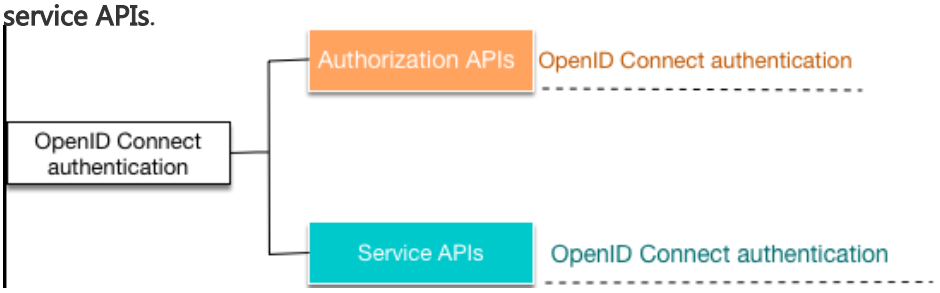
Verify the Time Stamp

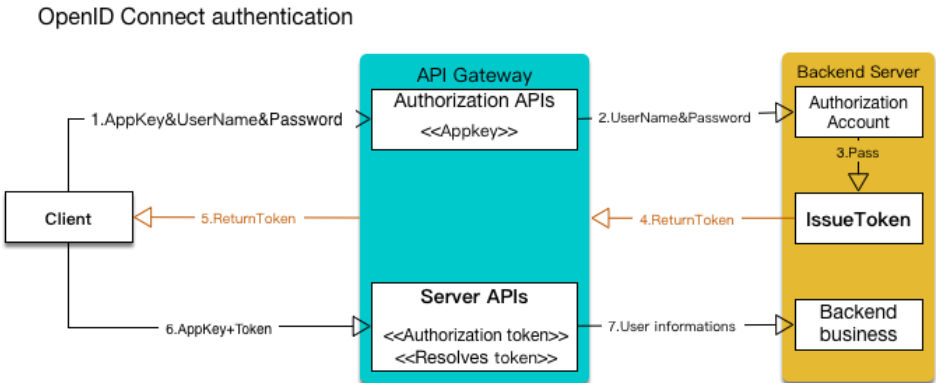
When the backend verifies the time stamp of the request, the system parameter **CaRequestHandleTime** is selectable in API definition and its value is the Greenwich mean time when the gateway receives the request.

OpenID Connect 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 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.

Implementation principle

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





- Authorization APIs: Interfaces used to issue a Token to the client. When configuring such APIs, you need to 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 need to 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

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

After receiving the request, the API gateway authenticates your **Appkey** first. 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 perform the following operations in advance:

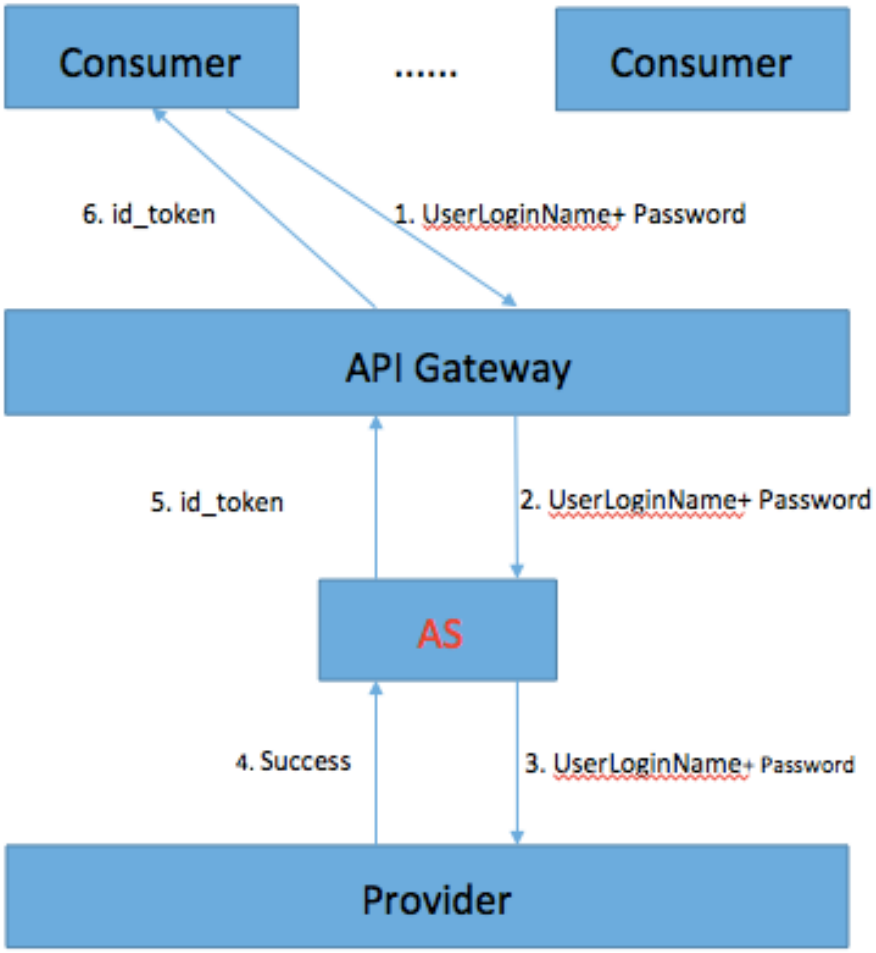
a. Opens the account system, allows the API gateway to authenticate the **user name/password** in the request, and issues the Token based on the gateway-provided encryption mode. For details, refer to **How to implement the AS module** below.
b. Defines the API in the API gateway. For details, refer to **Configure an API in the API gateway** below.
NOTE: The **user name/password** is extremely sensitive information, which is risky when being transmitted in plaintext. You are advised to 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 need to perform this step by yourself. For details about the method, refer to **Configure an API in the API gateway** below.



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).

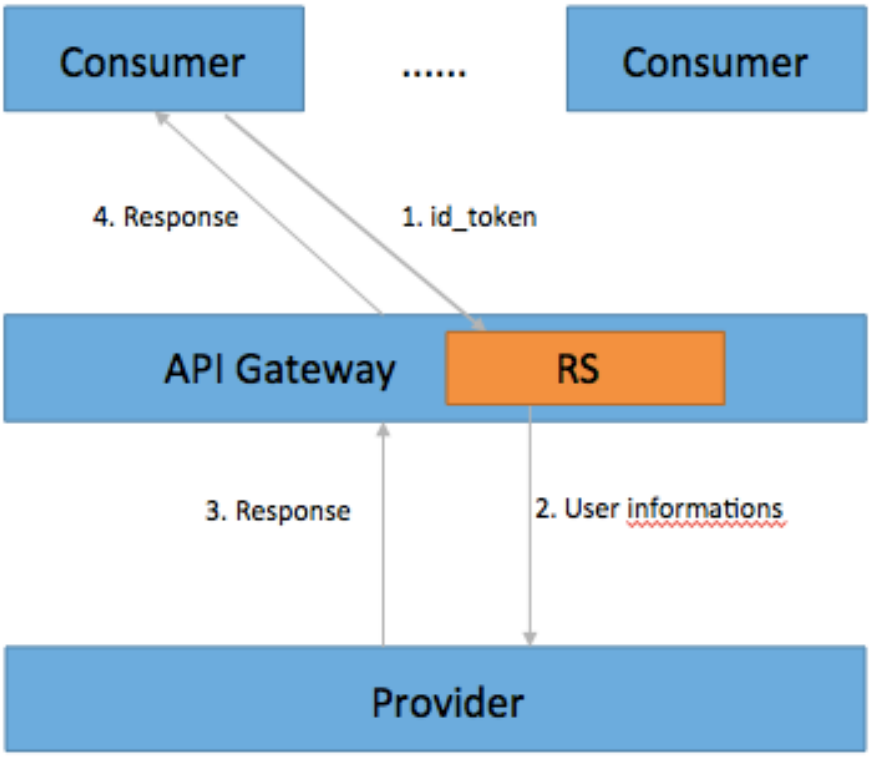
   The API gateway sends the id_token returned by the AS to the Consumer.

   NOTE: The AS does not need 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 details, refer to OpenID Connect Core 1.0.
- The KeyPair, keyId, and Claims are required to generate the id_token (for details about the Claims, refer to 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().getMostSignificantBits());
```

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 ensure 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":"oH5WunqaqIopfOFBz9RfBVVII
cmk0WDJagAcROKFiLJScQ8N\_nrexgbCMlu-dSCUWq7XMnp1ZSqw-XBS2-XEy4W4l2Q7rx3qDWY0cP8pY83hqxTZ6-
8GErJm\_0yOzR4WO4plIVVWt96-
mxn3ZgK8kmaeotkS0zS0pYMb4EEOxFFnGFqjCThuO2pimF0imxiEWw5WCdREz1v8RW72WdEfLpTLJEOpP1FsFyG3OI
DbTYOqowD1YQEf5Nk2TqN\_7pYrGRKsK3BPpw4s9aXHbGrpwsCRwYbKYbmeJst8MQ4AgcorE3NPmp-
E6RxA5jLQ4axXrwC0T458LIVhypWhDqejUw","e":"AQAB"}
```

privateKey:

```
{"kty":"RSA","kid":"67174182967979709913950471789226181721","alg":"ES256","n":"oH5WunqaqIopfOFBz9RfBVVII
cmk0WDJagAcROKFiLJScQ8N\_nrexgbCMlu-dSCUWq7XMnp1ZSqw-XBS2-XEy4W4l2Q7rx3qDWY0cP8pY83hqxTZ6-
8GErJm\_0yOzR4WO4plIVVWt96-
mxn3ZgK8kmaeotkS0zS0pYMb4EEOxFFnGFqjCThuO2pimF0imxiEWw5WCdREz1v8RW72WdEfLpTLJEOpP1FsFyG3OI
```

DbTYOqowD1YQEf5Nk2TqN_7pYrGRKsK3BPpw4s9aXHbGrpwsCRwYbKYbmeJst8MQ4AgcorE3NPmp-E6RxA5jLQ4axXrwC0T458LIVhypWhDqejUw","e":"AQAB","d":"aQsHnLnOK-1xxghw2KP5JTZyJZsiwt-ENFqqJfPUzmlYSCNAV4T39chKpkch2utd7hRtSN6Zo4NTnY8EzGQQb9yvunaiEbWUkPyJ6kM3RdlkkGLvVtp0sRwPCZ2EAYBlsMad9jkyrtmdC0rtf9jerzt3LMLC7XWbnpC3WAl8rsRDR1CGs_-u4sfZfttsaUbJDD9hD0q4NfLDCVOZoQ_8wkZxyWDAQGCe6GcCbu6N81fTp2CSVbiBj7DST_4x2NYUA2KG8vyZYcwviNTxQzk4iPfdN2YQz_9aMTZmmhVUGlmTvAjE5ebBqcqKAS0NfhOQHg2uR46eBKBy_OyVOLohsQ","p":"8Tdo3DCs-0t9JMtM0lYqPRP4wYJs37Rv6S-ygRui2MI_hadTY9I2A199JMYw7Fjke_wa3gqJLa98pbybdLWkrOxXbKEkwE4uc4-fuNjLbUTC5tqdM5-nXmpL887uREVYnk8FUzvWeXYTCNCb7OLw5l8yPJ1tR8aNcd0fJNDKh98","q":"qlRrGSTsZzBkDgDi1xlCoYvoM76cbmxrCUK-mc_kBRHfMjlHosxFUnAbxqIBE4eAJEKVfIJLQrHFvIDjQb3kM9ylmwMCu9f8u9DHrT8J7LSDlLqDaXuiM2oiKtW3bAaBPuiR7sVMFcuB5baCebHU487YymJCBTfeCZtFdi6c4w0","dp":"gVCROKonsjiQCG-s6X4j-saAL016jJsw-7QEYE6uiMHqR_6iJ_uD1V8Vuec-RxaItyc6SBsh24oeqsNoG7Ndaw7w912UVDwVjwJKQFCJDjU0v4oniItosKcPvM8M0TDUB1qZojuMCWWRYsJjNSWcvAQA7JoBAd-h6I8AqT39tcU","dq":"BckMQjRg2zhnjZo2Gjw_aSFJZ8iHo7CHCi98LdlD03BB9oC_kCYEDMLGDr8d7j3h-llQnoQGbmN_ZeGy1l7Oy3wpG9TEWQEDEpYK0jWb7rBK79hN8l1CqyBlvLK5oi-uYCaiHkwRQ4RACz9huyRxKLOz5VvlBixZnFXrzBHVPlk","qi":"M5NCVjSegf_KP8kQLAudXUZi_6X8T-owtsG_gB9xYVGnCsbHW8gccRocOY1Xa0KMotTWJl1AskCu-TZhOJmrdeGpvkdulwmbIcnjA_FgfIp4lAj4TCWmtRI6982hnC3XP2e-nf_z2XsPNiuOactY7W042D_cajyyX_tBEJaGOXM"}

Example of generating a KeyPair (Java)

```
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(JsonWebKey.OutputControlLevel.PUBLIC\_ONLY);
String privateKey = jwk.toJson(JsonWebKey.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);
ws.setPayload(claims.toJson());
PrivateKey privateKey = newRsaJsonWebKey(JsonUtil.parseJson(privateKeyText)).getPrivateKey();
jws.setKey(privateKey);
```

Use the JWS to obtain the value of the id_token.

Code example (Java)

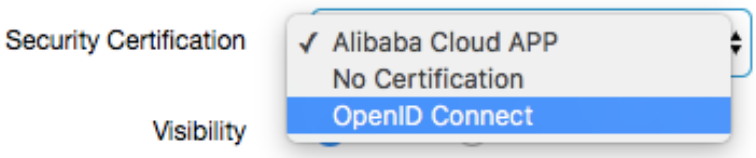
```
String idToken = jws.getCompactSerialization();
```

Example of a generated id_token:

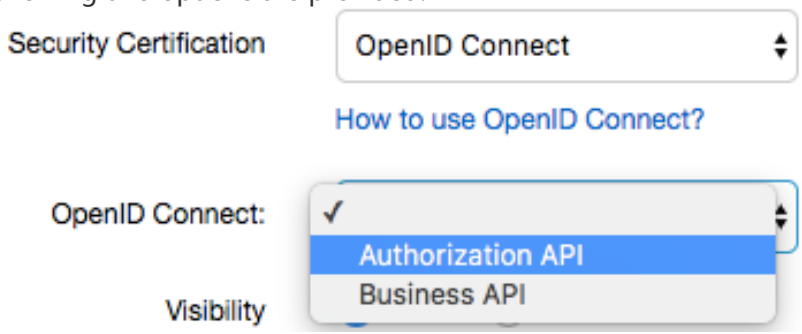
```
eyJhbGciOiJSUzI1NiIsImtpZCI6Ijg4NDgzNzI3NTU2OTI5MzI2NzAzMzA5OTA0MzUxMTg1ODE1NDg5In0.eyJ1c2VySWQiOiIzMzcwMTU0NDA2ODI1OTY4NjI3IiwidGFnTmFtZSI6ImNvbmFuVGVzdCIsImV4cCI6MTQ4MDU5Njg3OSwiYXVkIjoiQWxpX0FQSV9Vc2VyIiwianRpIjoiTm9DMFVVeW5xV0N0RUFEVjNoeEIydyIsImlhdCI6MTQ4MDU5MzI3OSwibmJmIjoxNDgwNTkzMjE5LCJzdWIiOiJ7ZGF0YU1hcD0ne3VzZXJJZD0zMzcwMTU0NDA2ODI1OTY4NjI3fScsIHN0YXR1c0NvZGU9JzAnLCBlcnJvcnM9J1tdJ30ifQ.V3rU2VCziSt6uTgdCktYRsIwkMEMsO_jUHNCCIW_Sp4qQ5ExjtwNt9h9mTGKFRujk2z1E0k36smWf9PbNGTZTWmSYN8rvcQqdsupcC6LU9r8jreA1Rw1CmmeWY4HsfBfeInr1wCFrEfZl6_QOtf3raKSK9AowhzEsnYRKAYuc297gmV8qlQdevAwU75qtg8j8ii3hZpJqTX67EteNCHZfhXn8wJjckl5sHz2xPPyMqj8CGRQ1wrZEHjUmNPw-unrUkt6neM0UrSqcjlrQ25L8PEL2TNs7nGVdl6iS7Nasbj8fsERMKcZbP2RFzOZfKJuaivD306cJIpQwxfS1u2bew
```

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.



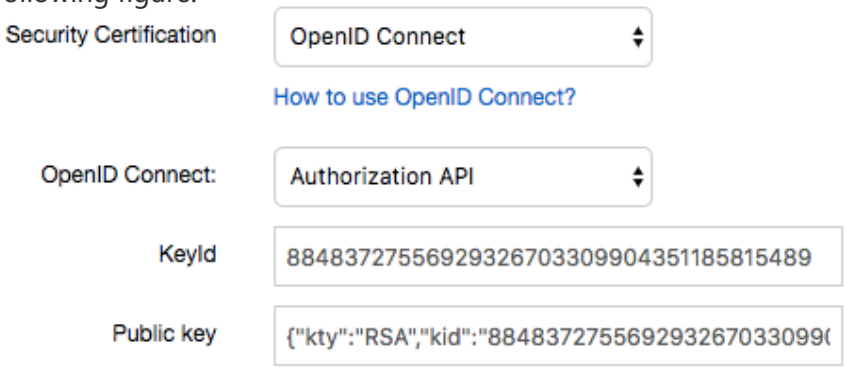
After selecting **OpenID Connect** for Security certification, set **OpenID Connect mode**. The following two options are provided.



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 need to configure the KeyId and publicKey, as shown in the following figure.



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

```
88483727556929326703309904351185815489
```

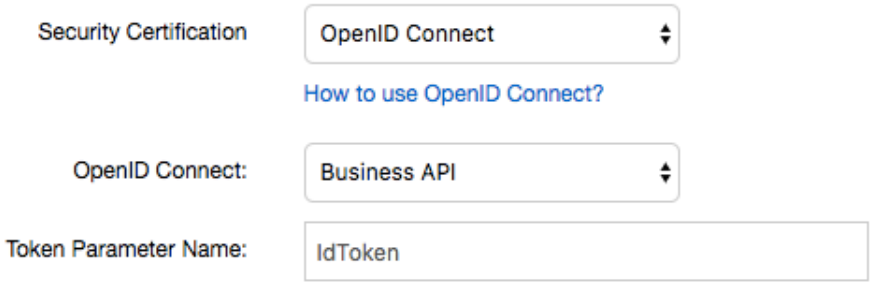
publicKey: Used to verify and resolve the Token, which is generated by the AS. For example:

```
{"kty":"RSA","kid":"88483727556929326703309904351185815489","alg":"ES256","n":"ie0IKvKLd7Y3izHcZemdDsVVXg5QtWtGF7XEkILnn66R2\_3a30DikqV409OVL7Hv0ElACgCaBLEgZeGHTcdLE1xxDTna8MMBnBNuMVghvFERCKh8uzpxlQsfcnFd5IFdJWj1x5Tscetrow6lA3h5zYx0rF5TkZzC4DclxgDmITRam0dsHBxr3uk9m9YYBz2mX0ehjY0px7vIo7hZH2J3gODEPorIZkk3x8GPdlaA4P9OFAO4au9-zcVQop9vLirxdwDedk2p-F9GP6UiQC9V2LTWqkVw\_oPBf9Rlh8Qdi19jA8SeCfzAxJZYlbOTK8dYAFAVEFsvXCFvdaxQefwWFw","e":"AQAB"}
```

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

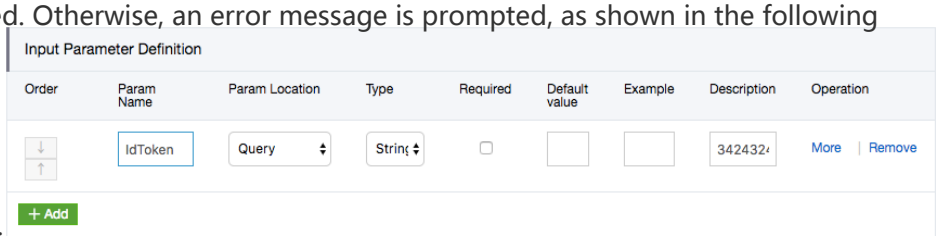
No matter creating an API or modifying an API, the configured KeyId and publicKey take effect only after the API is released.

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

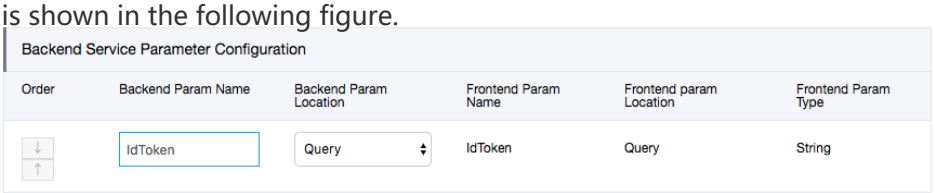


As shown in the preceding figure, the parameter corresponding to the Token is that sent to the id_token when the Consumer calls the API. The API gateway identifies, verifies, and resolves this parameter.

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



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.

The above is how to configure third-party authenticated OpenID Connect in the API gateway for your reference.

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, ensure that you have carefully read RAM help manual and API gateway API manual.
- **Skip this section if you do not have such service scenarios.**

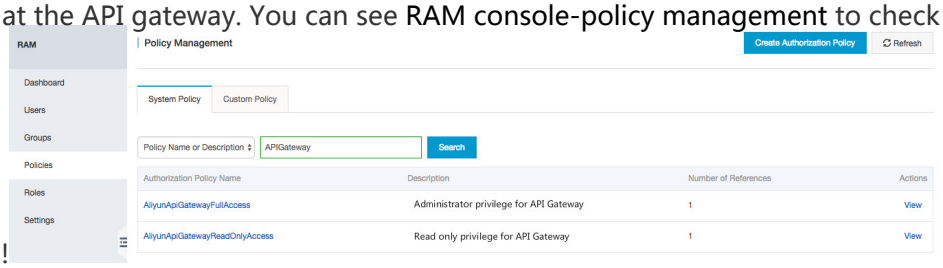
You can use the RAM console or API to add operations.

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.!



- 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

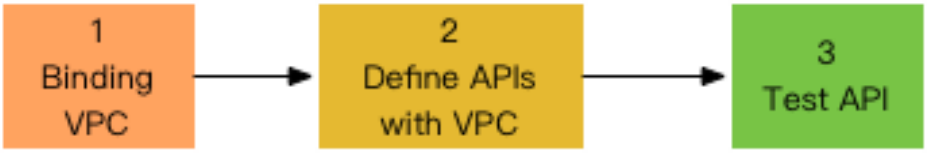
	/$groupId
DescribeHistoryApi	acs:apigateway:$regionid:$accountid:apigroup/$groupId
DescribeHistoryApis	acs:apigateway:$regionid:$accountid:apigroup/*
DescribeRulesByApi	acs:apigateway:$regionid:$accountid:group/$groupId
DescribeSecretKeys	acs:apigateway:$regionid:$accountid:secretkey/*
DescribeTrafficControls	acs:apigateway:$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:trafficc

	ontrol/$trafficcontrolId)
SetDomain	acs:apigateway:$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

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, ensure that you have understood how to use VPC.

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

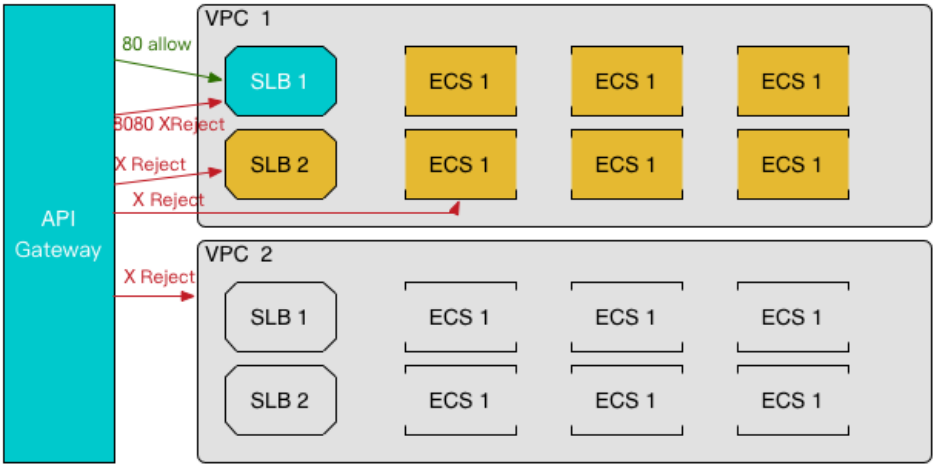


1 Authorize and bind a VPC instance

In a VPC environment, you need to authorize the API gateway so that it can access the service in your VPC. During authorization, you need to 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

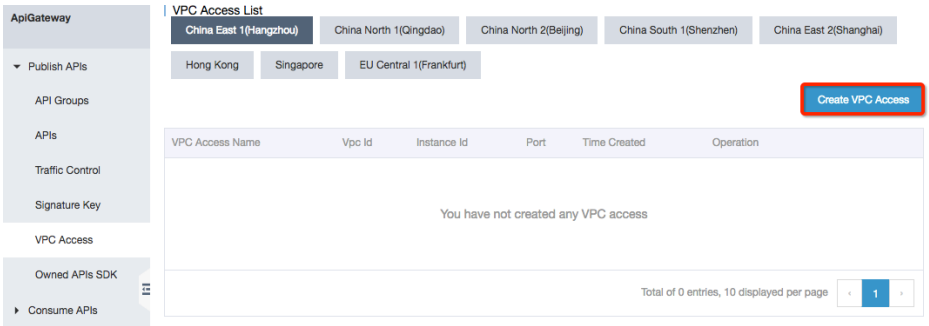
(1) Buy Server Load Balancer and ECS instances in the VPC environment and build the service. For details, refer to VPC user manual.

(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.

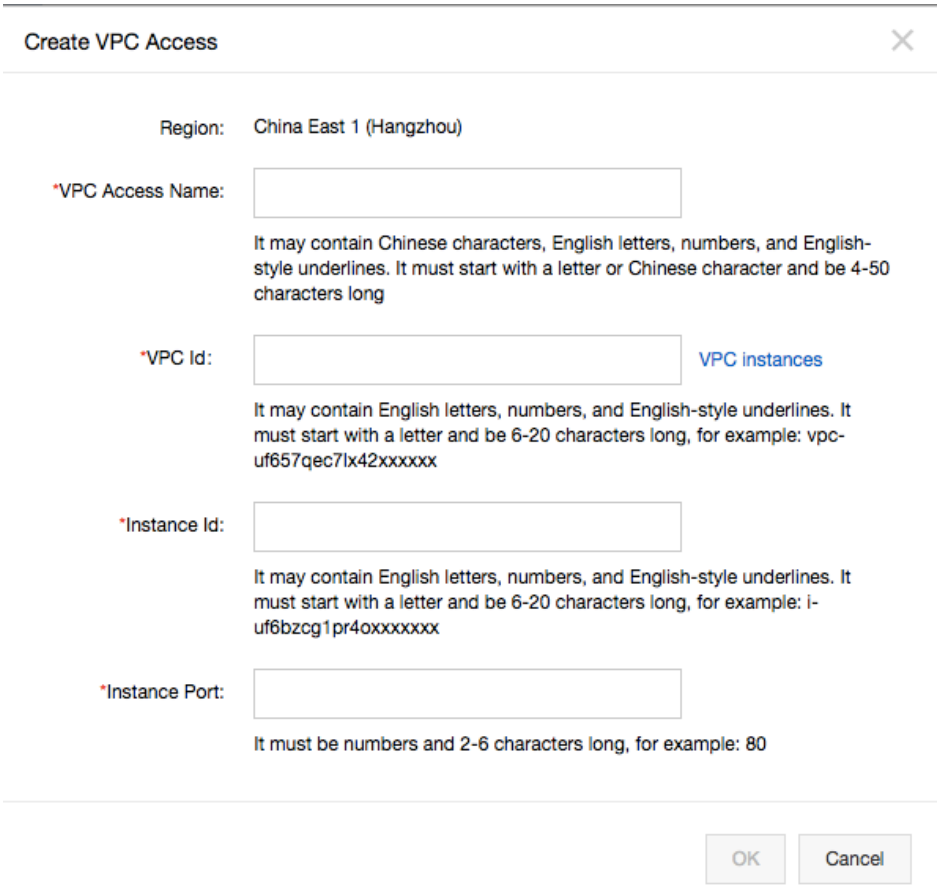
1.2 Authorize the API gateway for access

Choose "API Gateway Console" > "Open API" > "Authorize VPC".Click "Create Authorization".



Go to the authorization page and fill in corresponding information.

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



Click “OK” to complete the authorization.

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

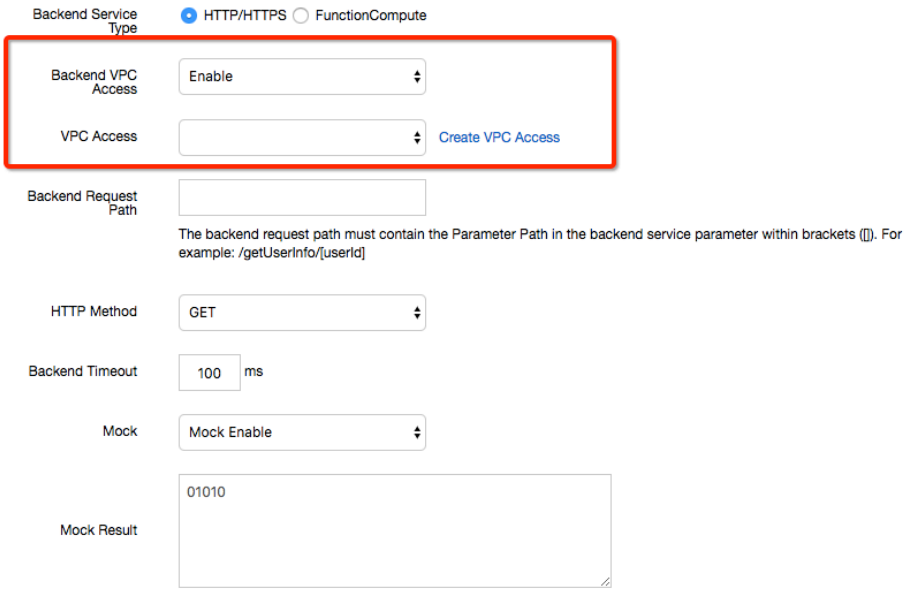
2 Create an API

The process for creating an API is the same as that for creating other APIs. For details, refer to **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.



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 need to 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
Hangzhou	Intranet inbound	100.104.13.0/24
Beijing	Intranet inbound	100.104.106.0/24
Shenzhen	Intranet inbound	100.104.8.0/24
Shanghai	Intranet inbound	100.104.8.0/24
Hong Kong	Intranet inbound	100.104.175.0/24
Singapore	Intranet inbound	100.104.175.0/24

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?

Ensure 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.

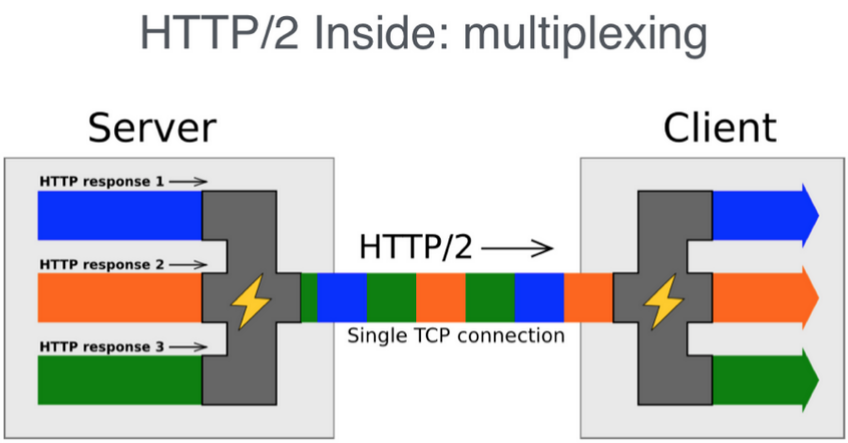
1. Security control authorization: Only the owner of the VPC can perform authorization.
2. Exclusive channel between the API gateway and VPC after authorization: Other persons cannot use this channel.
3. Authorization for the port of a certain resource: The gateway does not have the permission to access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

API Gateway supports HTTP 2.0

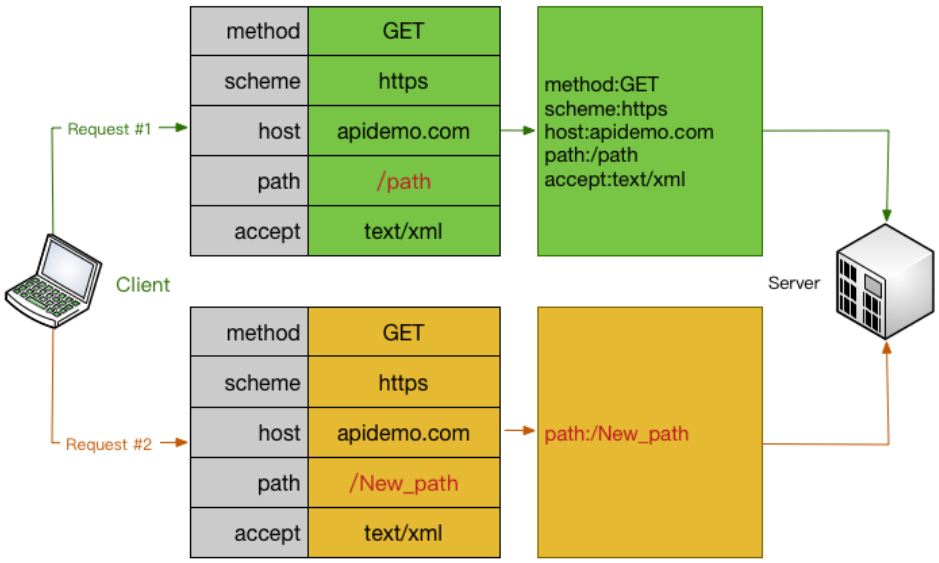
API Gateway supports new features of HTTP 2.0, multiplexing, and request header compression.

- 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.



- 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.

HTTPS is a protocol integrating HTTP and SSL. It encrypts information and data to ensure 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, perform the following steps:

Step 1. Prepare materials

Prepare the following materials:

- A self-owned controllable domain name
- An SSL certificate applied for this domain name

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

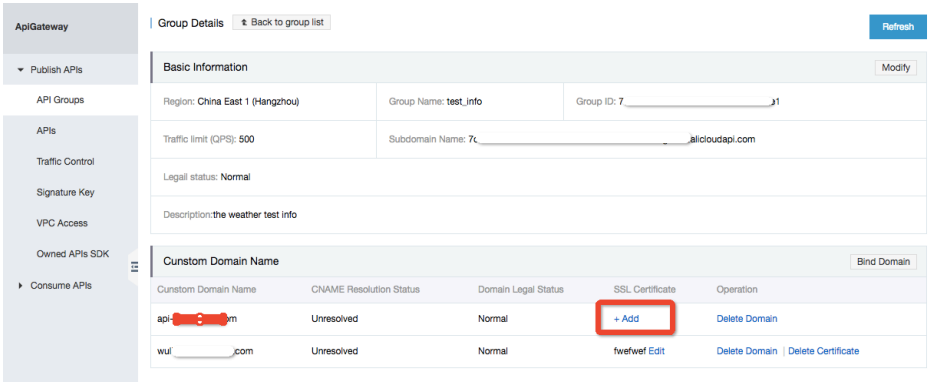
```
-----BEGIN CERTIFICATE-----
MIIFtDCCBJygAwIBAgIQRgWF1j00cozRl1pZ+ultKTANBgkqhkiG9w0BAQsFADBP
...
-----END CERTIFICATE-----
```

Step 2: Bind the SSL certificate

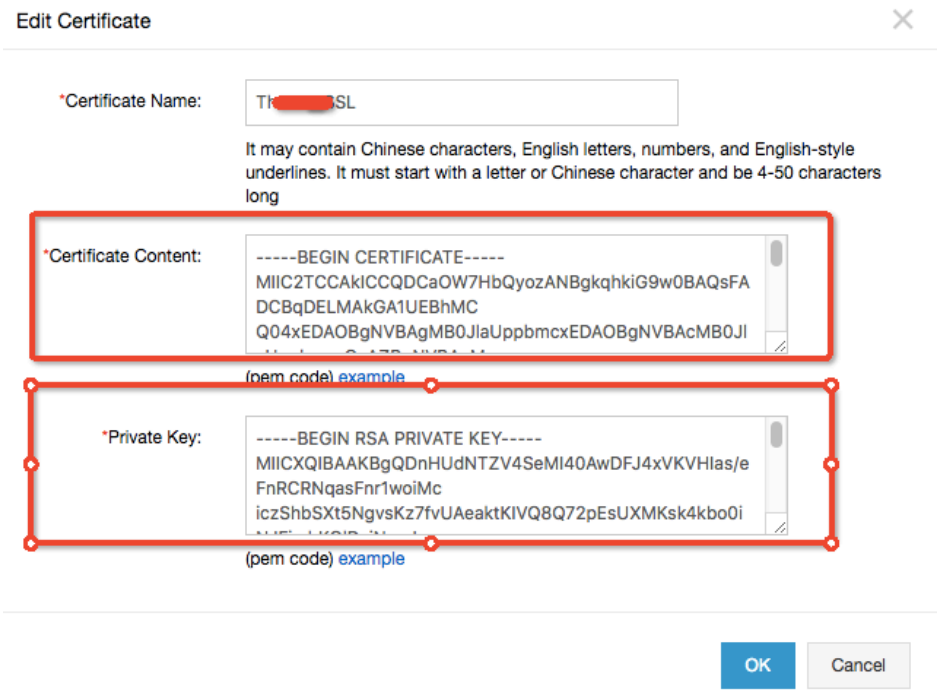
After preparing the preceding materials, log in to the API gateway console and choose “Open API”

> “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.



“Independent domain name” - Add an SSL certificate.

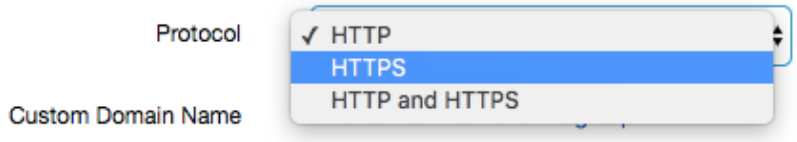


- Certificate name: Indicates the custom name for further identification.
- Certificate content: Indicates the complete content of the certificate. You need to copy all content in XXXXX.pem.
- Private key: Indicates the private key of the certificate. You need to copy the content in XXXXX.key.

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, you are advised to configure all APIs to support access over HTTPS.



You can choose "Open API" > "API list" to locate the corresponding API and choose "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.