

Alibaba Cloud

API Gateway Best Practices

Document Version: 20220614

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).
5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.
6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions









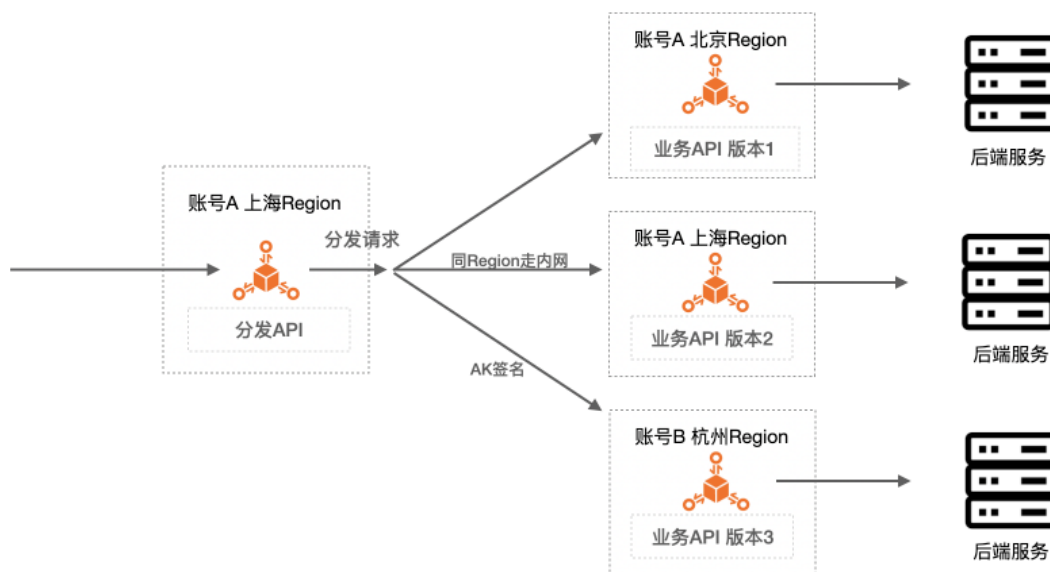
Style	Description	Example
 Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 Danger: Resetting will result in the loss of user configuration data.
 Warning	A warning notice indicates a situation that may cause major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

Table of Contents

1.Call by API Gateway	05
2.Integrate API Gateway with a CI/CD process based on Swagger	09
3.Best practices for implementing a canary release in API Gatew... ..	19

1.Call by API Gateway

APIs that are created in API Gateway can be called not only by the client but also by API Gateway. API Gateway can call the APIs across regions and call the APIs over the internal network in a region. API Gateway can also call an API across accounts by using an AccessKey pair of an authorized application to bind a backend signature plug-in of the APIGW_FRONTEND type. Before API Gateway calls an API, API Gateway uses the AccessKey pair to generate a signature and sends the signature to the API for authentication. This feature can be used in the following typical scenario: You create an API that is used to route requests. The API is bound with a backend routing plug-in and a backend signature plug-in. In the backend, the backend routing plug-in routes requests to other APIs based on the request parameters.

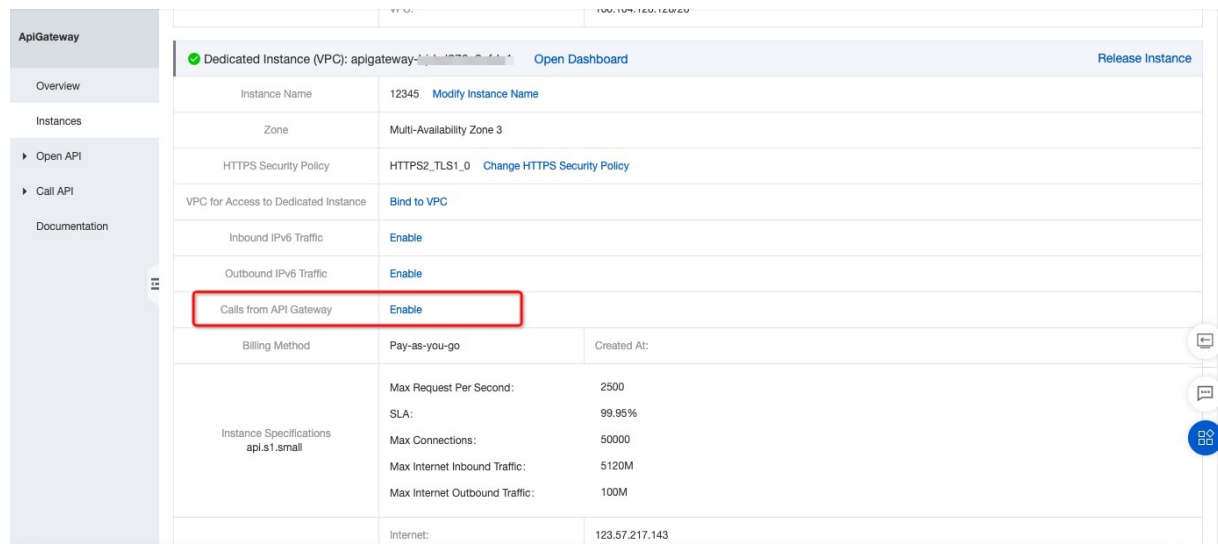


1. Example

1.1. Configure APIs

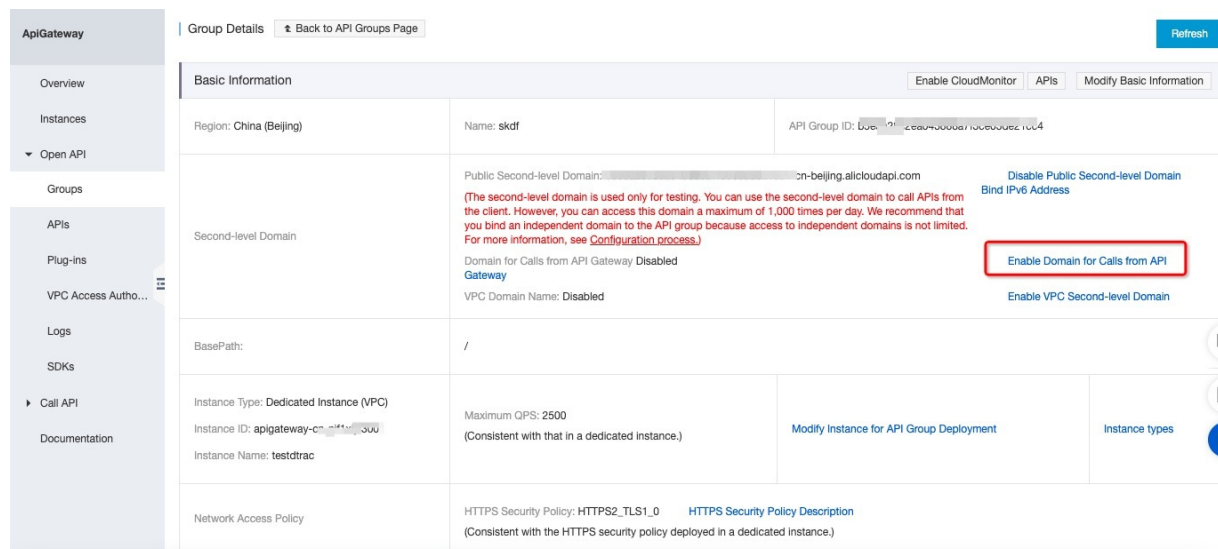
If you want API Gateway to call an API over the internal network, you must purchase an exclusive instance first. Then, you must migrate the API group to which the API belongs to the exclusive instance and manually generate an internal domain name for API calls in the API Gateway console.

1.1.1. Enable API calls over the internal network for the exclusive instance



1.1.2. Enable internal domain names that are used for API calls from API Gateway for API groups

Create two API groups and generate an internal domain name that is used for API calls from API Gateway for each API group.



For example, the following two internal domain names are generated:

17ff4c9189004a1d87b557606b767334-cn-huhehaote-inttranet.alicloudapi.com
 c6e984b2dd784c0fb843f7c2a8878b15-cn-huhehaote-inttranet.alicloudapi.com

1.1.3. Create an API in each API group

Create an API in each API group. Applications must be authorized before they can call the two APIs. The following example shows the attributes of the two APIs:

- API1: Method: Get Path: /business1 Backend service address:

```
http://backend1.alicloudapi.com:8080/business1
```

- API2 Method: Get Path: /business2 Backend service address:

```
http://backend2.alicloudapi.com:8080/business2
```

1.1.4. Grant the permissions on the two APIs

Grant the permissions on the two APIs to an application. In this example, the application has the following AccessKey pair: AccessKey ID:TESTKEY AccessKey secret:TESTSECRET

1.2. Configure an API that is used to route requests

1.2.1. Create an API that is used to route requests

Create an API that is used to route requests. The API can be called anonymously. Set the request method to Get and the path of the API to /distributeAPI. In this example, the domain name of the API group to which the API belongs is 17ff4c9189004a1d87b557606b767334-cn-huhehaote.alicloudapi.com.

1.2.2. Create and bind a backend routing plug-in

Create a backend routing plug-in and bind the backend routing plug-in to the API that is used to route requests.

```
---
parameters:
  target: "Query:target"
routes:
- name: backend1
  condition: "$target = 'resource1'"
  backend:
    type: "HTTP"
    address: "17ff4c9189004a1d87b557606b767334-cn-huhehaote-intranet.alicloudapi.com"
    path: "/business1"
- name: backend2
  condition: "$target = 'resource2'"
  backend:
    type: "HTTP"
    address: "c6e984b2dd784c0fb843f7c2a8878b15-cn-huhehaote-intranet.alicloudapi.com"
    path: "/business2"
```

After the API is bound with the backend routing plug-in, the API routes a request based on the preceding configurations. If the value of the request parameter target is resource1, the API sends an HTTP request whose path is /business1 to 17ff4c9189004a1d87b557606b767334-cn-huhehaote-intranet.alicloudapi.com. If the value is resource2, an HTTP request is sent based on the preceding configurations.

1.2.3. Create and bind a backend signature plug-in

Create a backend signature plug-in and bind the backend signature plug-in to the API that is used to route requests.

```
---
type: APIGW_FRONTEND
key: TESTKEY
secret: TESTSECRET
signatureMethod: HmacSHA256
```

After the API is bound with the backend signature plug-in, the API generates a signature based on the content of a request and the frontend signature algorithm of API Gateway. Then, the API includes the signature in the request and sends the request to the backend.

2. Call the API that is used to route requests

Before you call the API that is used to route requests, make sure that all the created APIs are published to the production environment. Then, you can run the following commands to perform testing:

```
curl 'http://17ff4c9189004a1d87b557606b767334-cn-huhehaote.alicloudapi.com/distributeAPI?target=resource1' -i
```

Request sent to the backend:

```
GET /business1 HTTP/1.1
User-Agent: curl/7.64.1
Via: 0045e52ee3a8400b8501b4c449b28779
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Forwarded-Proto: http
X-Forwarded-For: 106.1.1.1, 127.0.0.1
Host: backend1.alicloudapi.com:8080
X-Ca-Request-Id: 23853B41-C54D-45E9-8C43-EE4C1E8A7889
Via: bc48a42a3d17408b991b0bb4d18c23c0
```

```
curl 'http://17ff4c9189004a1d87b557606b767334-cn-huhehaote.alicloudapi.com/distributeAPI?target=resource2' -i
```

Request sent to the backend:

```
GET /business2 HTTP/1.1
User-Agent: curl/7.64.1
Via: 0045e52ee3a8400b8501b4c449b28779
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Forwarded-Proto: http
X-Forwarded-For: 106.1.1.1, 127.0.0.1
Host: backend2.alicloudapi.com:8080
X-Ca-Request-Id: AFD529D2-9B24-437E-8CEC-897E0BCD8B2F
Via: bc48a42a3d17408b991b0bb4d18c23c0
```


2. Integrate API Gateway with a CI/CD process based on Swagger

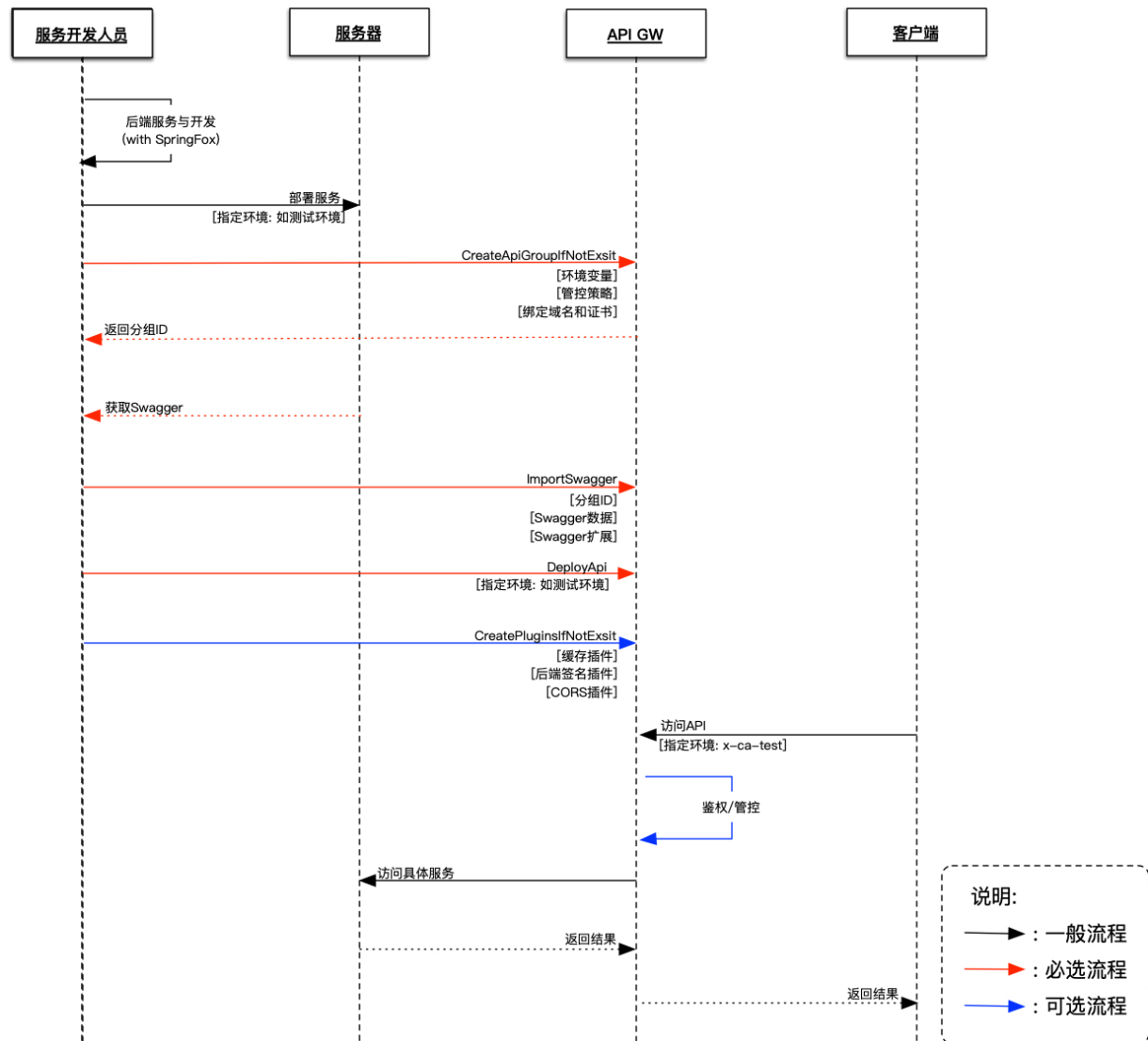
This topic describes how to integrate API Gateway with a continuous integration and continuous delivery (CI/CD) process based on Swagger. You can implement automatic API creation and updates by using the ImportSwagger operation provided by API Gateway.

1. Overview

API Gateway provides a comprehensive set of API operations for you to manage APIs that are hosted in the API Gateway console. The core of the integration between API Gateway and a CI/CD process is to encapsulate and integrate the management API operations provided by API Gateway.

The following figure shows a general CI/CD process integrated with API Gateway. This process consists of the following steps:

- Step 1: Obtain the Swagger specification from your API business code.
- Step 2: Create an API group.
- Step 3: Import the API definitions in the Swagger specification by using OpenAPI Explorer and publish the APIs to different environments.
- Step 4: Complete additional configurations for the APIs.



Obtain the Swagger specification

You can integrate an API doc framework into your backend service during business development. SpringFox is used as an example in this topic. The API doc framework is easy to access because it is non-intrusive to the original code. When the API doc framework starts with the backend services, a Swagger acquisition URL is generated. API Gateway accesses the URL to automatically obtain the Swagger specification.

Create an API group

You can manually create an API group in the [API Gateway console](#) or by calling the CreateApiGroup operation. After you create an API group, you can bind an independent domain name and a certificate to the API group.

Import and publish APIs by using OpenAPI Explorer

You can call the ImportSwagger operation of API Gateway and specify the required parameters based on the obtained Swagger specification to create APIs. After you create APIs in the API Gateway console, you can publish the APIs to different environments based on your business requirements. After the APIs are published, you can access the related services.

Perform additional configurations

API Gateway provides a variety of [Overview](#) to satisfy your API management requirements. The plug-ins include the throttling plug-in, IP address-based access control plug-in, CORS plug-in, and JWT plug-in. You can configure the plug-ins based on your business requirements to enhance the authentication and control capabilities of the related APIs.

2. Example

This section describes the sample code in a basic CI/CD scenario. You can modify the code based on your business requirements.

2.1 Sample code

[alibabacloud-cloudapi-java-cicd-demo](#)

2.2 References and technologies

This example involves the following references and technologies.

2.2.1 References

1. [Import Swagger files to create APIs](#): This topic describes how to configure a Swagger file that can be imported into API Gateway. API Gateway provides Swagger extensions to allow you to import both standard and custom Swagger files.
2. [Import Swagger files to create APIs](#): This topic describes the ImportSwagger operation. This operation involves the following parameters:
 - `groupId`: the ID of the API group to which the APIs you want to import belong. The APIs are imported by using Swagger.
 - `data`: the Swagger file.
 - `globalCondition`: custom configurations that you add based on your business requirements. For example, you can add custom backend service configurations by using the `x-apigateway-backend-info` label.

2.2.2 SpringFox

This topic provides a backend service that uses the SpringFox library to demonstrate how to synchronize APIs from a backend service to API Gateway by using Swagger. [SpringFox](#) is an open-source API doc framework. It can conveniently form a Swagger specification by adding annotations to a Controller operation of the backend service. This framework also generates a URL for API Gateway to obtain the Swagger specification when the backend service is started.

2.3 Automatic generation of a Swagger specification

A minimal web service based on Spring Boot is created on the server. This service shows how to generate Swagger based on existing services.

2.3.1 Dependent libraries

```
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger2</artifactId>
  <version>2.9.2</version>
</dependency>
```

2.3.2 SpringFox configurations

SpringFox is a framework that is ideal for your original services. To access this framework, you need only to add a configuration class of SpringFox. You do not need to modify the existing interfaces or model classes. After the service is started, this framework automatically parses the annotations in the Spring Boot web framework to generate service information required by Swagger. At the same time, this framework parses the classes referenced by the service to generate the model information required by Swagger.

Configuration items of SpringFox

The following sample code shows the core settings of SpringFox:

- Document format: Swagger2
- Name of the package used to generate the API documentation

```
@Configuration
@EnableSwagger2
public class SpringFoxConfig {
    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
            .select()
            .apis(RequestHandlerSelectors.basePackage("com.example.swagger.controller"))
        )
            .paths(PathSelectors.any())
            .build();
    }
}
```

View Swagger information

After a service is started, you can visit <http://localhost:8080/swagger-ui.html> to open the Swagger UI and use the Swagger features. For more information, see swagger.io at the official website of Swagger.

The following figure shows that the service interfaces and model classes in the sample code are automatically parsed.

Api Documentation ^{1.0}

[Base URL: localhost:8080/]

<http://localhost:8080/v2/api-docs>

Api Documentation

[Terms of service](#)

[Apache 2.0](#)

user-controller User Controller

GET /user getUser

Models

Company >

User >

Obtain the Swagger specification

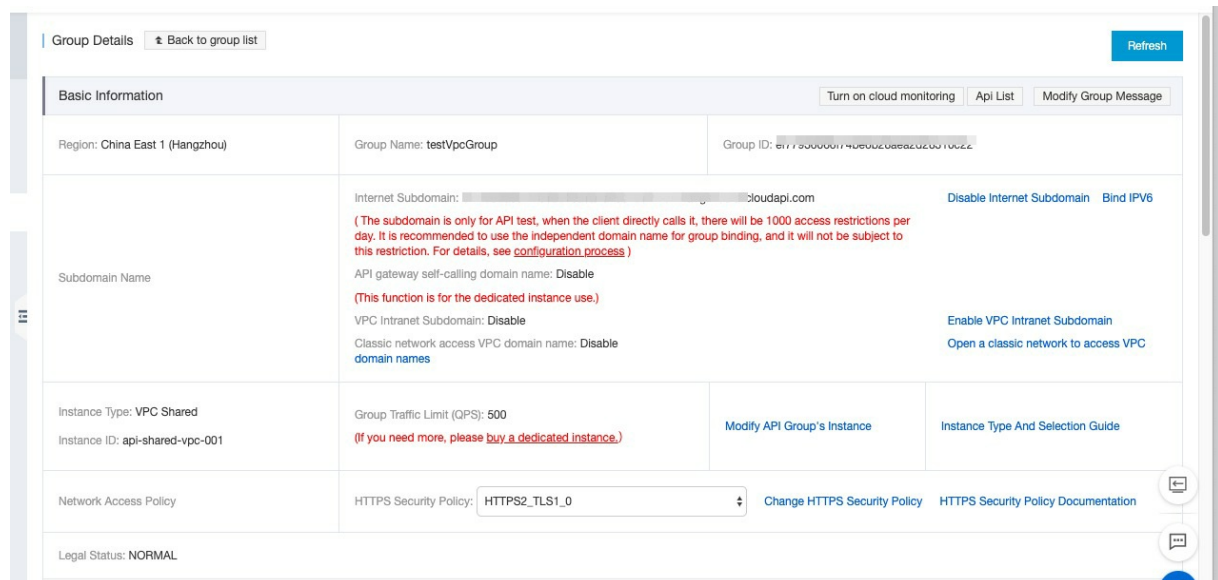
The URL from which the Swagger specification can be obtained is required if you want to integrate a CI/CD process with API Gateway. After a service is started, you can obtain the Swagger specification from the /v2/api-docs URL. `http://localhost:8080/v2/api-docs` [(`http://localhost:8080/v2/api-docs`)]

2.4 Create an API group

You can use one of the following methods to create an API group.

2.4.1 Create an API group in the API Gateway console

Log on to the [API Gateway](#) console. In the left-side navigation pane, choose Publish APIs > API Groups. On the Group List page, click Create Group. After you create an API group, you can go to the group details page to view the ID and other related information of this group.



2.4.2 Create an API group by using an API operation

Call the CreateApiGroup operation provided by API Gateway to create an API group. In addition, you can bind an independent domain name and certificate to the group by using other operations.

2.5 Use of the ImportSwagger operation

To integrate API Gateway and a CI/CD process, take note of the following key points:

- Whether a URL is available for API Gateway to obtain Swagger specification
- How to import standard Swagger files to API Gateway

As described in section 2.2.2, the Swagger files that describe all interfaces and model classes of backend services are obtained. This section focuses on how to import standard Swagger files to API Gateway.

2.5.1 Swagger extensions based on API Gateway

API Gateway allows you to import native Swagger files. This facilitates the implementation of the features provided by API Gateway. This way, API Gateway automatically creates Swagger-based APIs and the backend service runs in mock mode by default.

However, in most cases, the APIs created in the API Gateway console are required to access the backend service in CI/CD scenarios. Therefore, you must configure the backend service. Native Swagger does not contain the configuration information. To resolve this issue, API Gateway provides Swagger extensions. For more information, see [Import Swagger files to create APIs](#).

Swagger extensions include the information that API Gateway requires or the special configurations, such as backend service configurations, backend parameter configurations, and authentication methods. These extensions can be configured for all APIs at the same time or for each API individually. The use of native Swagger and Swagger extensions ensures that the API configurations meet the requirements of your production environment.

The `x-apigateway-backend-info` extension is used in this example. In this example, the backend service is HTTP. Assume that the domain name is `www.aliyun.com`. You can also configure the backend service address by using *#environment variables*.

The following example demonstrates how to configure this extension when an HTTP backend service is used. For more information, see [Import Swagger files to create APIs](#). The type and address parameters are required. The default timeout period is 10,000 ms.

```
x-aliyun-apigateway-backend:
  type: HTTP
  address: 'http://www.aliyun.com'
  path: '/builtin/echo'
  method: get
  timeout: 10000
```

2.5.2 Import the native Swagger file by using the ImportSwagger operation

The ImportSwagger operation is used to import native Swagger files to API Gateway. This operation has the following core parameters:

- `groupId`: the ID of the API group to which the APIs you want to import belong.
- `data`: the native Swagger file.
- `globalCondition`: custom configurations, such as the general backend service address.

Maven dependencies

```
<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-core</artifactId>
  <version>4.5.0</version>
</dependency>
<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-cloudapi</artifactId>
  <version>4.9.3</version>
</dependency>
```

Obtain the native Swagger file

After you apply the SpringFox framework and complete the related configurations, the `/v2/api-docs` URL is automatically generated when the service is started. API Gateway can obtain Swagger information that describes the backend service from the URL.

```
private String getSwaggerData() {
    HttpURLConnection connection = null;
    ...
    ...
    String result = null;
    try {
        URL url = new URL(swaggerApiDocUrl);
        connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod("GET");
        connection.connect();

        ...

        ...
    } catch (Exception e) {
        e.printStackTrace();
    } finally {

        ...

        connection.disconnect();
    }
    return result;
} ...
    connection.disconnect();
}
return result;
}
```

Configure Swagger extensions based on API Gateway

To create APIs by importing a Swagger file, you must correctly configure Swagger extensions. The `globalCondition` parameter provided by the `ImportSwagger` operation is used to pass the configurations of the Swagger extensions. The value of the `globalCondition` parameter is a JSON string.

- **key:** the name of a Swagger extension, for example, `x-apigateway-backend-in`.
- **value:** the value of a Swagger extension.

The following sample code demonstrates how to configure the general backend service address of APIs that are imported to API Gateway to enable the APIs to access their respective backend services.

An HTTP backend service is used in this example. The `SwaggerBackendInfoBase` class is created to configure the backend service information based on the description of the HTTP backend service.


```

public class SwaggerBackendInfoBase {
    private String type;
    private String address;
    public String getType() {
        return type;
    }
    public void setType(String type) {
        this.type = type;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
}

```

Set the `globalCondition` parameter based on the backend service information.

```

// Configure the backend service information required by API Gateway.
SwaggerBackendInfoBase info = new SwaggerBackendInfoBase();
info.setType("HTTP");
info.setAddress("http://www.aliyun.com");
// Add the custom backend service information by using the x-apigateway-backend
-info extension.
Map<String, String> globalCondition = new HashMap<>();
globalCondition.put("x-aliyun-apigateway-backend", JSON.toJSONString(info));

```

ImportSwagger

The operations, such as creating an API group, obtaining native Swagger, and adding the `GlobalCondition` parameter, have been completed. You can call the `ImportSwagger` operation provided by API Gateway to import API definitions and view the import results in the [API Gateway console](#).

The screenshot shows the 'Definition' page in the API Gateway console. At the top, there is a breadcrumb 'Definition' and a 'Back to API list' link. On the right, there are buttons for 'Clone', 'Edit', 'Deploy', and 'Undeploy'. The main content area is divided into three sections: 'Name And Description', 'Basic Request Definition', and 'Request Input Parameter Information'.

Name And Description	
Group: vpc1	API Name: ApiDescription
Security Certification: Alibaba Cloud APP	Whether to allow the cloud market to be listed: Yes
Whether to allow public network access: Yes	Whether to set to prevent replay attacks: No
Signature Method: HmacSHA256	AppCode Certification: Open after putting on cloud market
Description:	

Basic Request Definition	
Request Type: NOTIFY(WEB_SOCKET)	
Path: /regions	Protocol: HTTP
HTTP Method: GET	Request Mode: Request Parameter Mapping(Filter Unknown Parameters)

Request Input Parameter Information	

2.6 Publish APIs

You can go to the API List page for a specific API group, select the APIs that you want to publish, and click Deploy. Then, the APIs take effect.

API List

API Name: Enter the API name Search Tags

Create API Import Swagger Create Data Service APIs

Filters: Group Name:testGroup12123 Clear

<input checked="" type="checkbox"/>	API Name	Tag	Visibility	Group	Description	Last Modified	Stage (All)	Operation
<input checked="" type="checkbox"/>	apitest11212		Private	testGroup12123		Sep 24,2020 10:44:44	Release Pre Test	Deploy Debug More
<input checked="" type="checkbox"/>	testAPI		Private	testGroup12123		Sep 24,2020 10:25:30	Release (Running) Pre Test	Deploy Debug More
<input checked="" type="checkbox"/>	testAPI12		Private	testGroup12123		Oct 29,2020 11:08:02	Release (Running) Pre Test	Deploy Debug More

Export Swagger Authorize Deploy Undeploy Delete

Total of 3 entries, 10 displayed per page 10 1

3. Summary

This topic describes how to synchronize the APIs of your backend service to API Gateway. API Gateway allows you to focus on the development of backend services. It provides general features such as authentication and throttling. The ImportSwagger operation provided by API Gateway associates backend services with API Gateway. This significantly simplifies the configuration process for API Gateway. This operation also makes it possible to integrate the configurations and management of API Gateway into a CI/CD process.

3. Best practices for implementing a canary release in API Gateway

This topic describes how to implement a canary release for the new version of APIs in API Gateway before these APIs are officially published. An A/B test is a good practice to implement a canary release. You can configure the routing plug-in to control the publishing scope of the APIs.

1. Overview

A canary release allows you to upgrade APIs with the minimum impacts on users. If you want to upgrade APIs in API Gateway, you can publish the APIs of the new version in a small scope for verification before you officially publish them. This ensures stability of your online services and limits the impacts of the upgrade to a small number of users. After you verify the functions, performance, and stability of the APIs based on collected user experience data, you can publish these APIs to all users.

In most scenarios of canary releases and A/B tests, users are identified based on their information. Specifically, the routing plug-in provided by API Gateway identifies users based on a parameter in HTTP requests that call the APIs, for example, the `userId` parameter. Then, the plug-in distributes the requests to different backend services based on configured logic conditions.

2. Typical scenarios

You can configure request distribution conditions in the routing plug-in based on the following scenarios:

- Distribute requests based on the apps authorized by API Gateway. Different apps identify different API callers. You can configure the routing plug-in to distribute the requests of different apps to different backend services. For example, an API is authorized to multiple apps. When it is upgraded in one of the backend services, you can configure the routing plug-in based on section 3.1 to distribute API requests from two of these apps to the upgraded backend service. This allows you to limit impacts on online services to a small scope.
- Distribute requests based on a JSON Web Token (JWT) claim. If a JWT is bound to an API, API Gateway can parse an unencrypted claim that identifies users from the JWT token, such as `userId`. Then, the routing plug-in distributes API requests to different backend services based on the claim. For more information, see

JWT authentication

You can bind a JWT to the API and configure the routing plug-in based on section 3.2. In the example in section 3.2, 20% of the requests are distributed to the upgraded backend services based on the last number in the value of the `userId` claim.

- Distribute requests based on a user parameter in HTTP requests. If the query, form, or header of the requests carries a parameter that can be used to identify users, the routing plug-in distributes the requests to different backend services based on this parameter. For example, the client version of some users is not the latest, and you want to prompt them to upgrade the client. You can configure the routing plug-in based on section 3.3.

- Distribute requests based on source IP addresses. The routing plug-in can distribute requests to different backend services based on the source IP addresses. For example, you want to use the same frontend code in both the test and production environments. You can configure the routing plug-in based on section 3.4 to direct users from the test and production environments to different backend services.

3. Configuration

This section describes how to configure the routing plug-in for the scenarios in section 2. After you bind the configured plug-in to the API, requests that match the conditions are distributed to the backend service specified by the plug-in. Requests that do not match the conditions are forwarded to other backend services. If you want to direct users to more than two types of backend services, you can configure multiple routes in the [Plug-ins of the Routing type](#). You must define parameters used in conditional expressions of the plug-in based on [Use parameters and conditional expressions](#).

3.1. Distribute requests based on the apps authorized by API Gateway. If you want to distribute requests that call the API authorized to some apps to the test backend service, configure routing policies based on the `CaAppId` parameter.

```
---
routes:
- name: appGrey
  condition: "$CaAppId = 4534463 or $CaAppId = 86734572"
  backend:
    type: HTTP
    address: "http://example-test.alicloudapi.com:8080"
    path: "/web/xxx"
    method: GET
    timeout: 7000
```

3.2. Distribute requests based on a JWT claim. Assume that you have specified a JWT claim `userId`, and the last number in the value of `userId` is evenly distributed. To distribute 20% of users to the test backend service, perform the following steps:

- Bind a JWT to the test API.

```

---
parameter: X-Token          # Obtain the JWT from the specified API parameter.
parameterLocation: header  # The location to obtain the JWT. Valid values: query and header
                           . If the API works in mapping mode, this parameter is optional. If the API works in passthrough mode, this parameter is required.
claimParameters:           # Map the JWT claim to a backend parameter.
- claimName: userId         # The name of the claim. The claim can be private or public.
  parameterName: userId     # The name of the parameter mapped to the claim.
  location: query           # The location of the mapped parameter. Valid values: query, header, path, and formData.
#
# The public key in the JSON Web Key (JWK).
jwk:
  kty: RSA
  e: AQAB
  kid: showJwt
  alg: RS256
  n: gNHI8tm3lnsdCi09SrBPs9-Oau7Z1SFhIEOT2h5AJ49FSJA0XEyU4OadtV70BLIEy94dzcUK8f0e477AVoU00
  RZdcXjztFtpJnAlKtrzn9zAmKcXb2IuKXrBKkQStcKqoSbBlR84mDElp_gxfNqpmoLy0q08rkmjhlutd8E_S4QMDDaF
  tQ68ggJcDY-oX5FSiVidKNrKagEzQKpk5SgJFE8wpJOkW-YKouqLsL5lFyqnkg7J3MvDqEBKqgiCY-zXYaxnkLNfkr
  At7jTe4b4a2PiKD0-bHIZwzd2NVhuLGwx4pB1tFL51E-KeewZhTsoUbQ3v_ZerZ2_630WOH7IWQ

```

- Add `userId` as a custom claim to the payload of the JWT. For more information, see

[Implement OpenID Connect authentication by using API Gateway](#)

Add an `X-Token` field to the header of requests that call the API. Set the value of `X-Token` to the ID token that is generated when the private key in the JWK is used to validate the JWT signature.

- Configure the routing plug-in and bind it to the API. The plug-in can distribute requests in which the value of the `userId` parameter ends with 0 and 1 to the test backend service. These users account for 20% of the total.

```

---
parameters:
  userId: "Token:userId"
routes:
- name: userIdGrey
  condition: "$userId like '%0' or $userId like '%1'"
  backend:
    type: HTTP
    address: "http://example-test.alicloudapi.com:8080"
    path: "/web/xxx"
    method: GET
    timeout: 7000

```

3.3. Distribute requests based on the client version in HTTP requests. The following example shows an HTTP request to call the test API:

```
http://example-cn-hangzhou.alicloudapi.com/testJwtShow?clientVersion=1.9
```

Configure the routing plug-in to distribute requests in which the value of the `clientVersion` parameter is smaller than 2.0.5 to the test backend service.

```
---
parameters:
  clientVersion: "Query:clientVersion"
routes:
- name: oldVersionClient
  condition: "$clientVersion < '2.0.5'"
  backend:
    type: "MOCK"
    statusCode: 400
```

3.4. Distribute requests based on source IP addresses. If you want to direct internal users to the test environment and external users to the production environment, you can distribute requests based on the users' source IP addresses. In the following configuration, the routing plug-in distributes requests from 106.11.31.0/24 to the specified backend service based on the system parameter CaClientIp in API Gateway:

```
routes:
- name: InternalTesting
  condition: "$CaClientIp in_cidr '106.11.31.0/24'"
  backend:
    type: HTTP
    address: "http://example-test.alicloudapi.com:8080"
    path: "/web/xxx"
    method: GET
    timeout: 7000
```