ALIBABA CLOUD

Alibaba Cloud

Apsara Stack Enterprise

User Guide - Middleware and Enterprise Applications

Product Version: 2105, Internal: V3.14.0

Document Version: 20210826

Alibaba Cloud

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 `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.API Gateway

1.1. What is API Gateway?

API Gateway provides a comprehensive suite of API hosting services that help you share capabilities, services, and data with partners in the form of APIs.

- API Gateway provides multiple security mechanisms to secure APIs and reduce the risks arising from open APIs. These mechanisms include protection against replay attacks, request encryption, identity authentication, permission management, and throttling.
- API Gateway provides API lifecycle management that allows you to define, publish, and unpublish APIs. This improves API management and iteration efficiency.

API Gateway allows enterprises to reuse and share their capabilities with each other so that they can focus on their core business.

API Gateway

1.2. Log on to the API Gateway console

This topic describes how to log on to the API Gateway console.

Prerequisites

- The URL of the Apsara Uni-manager Management Console is obtained from the deployment personnel before you log on to the Apsara Uni-manager Management Console.
- A browser is available. We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL of the Apsara Uni-manager Management Console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password that you can use to log on to the console from the operations administrator.

> **Note** When you log on to the Apsara Uni-manager Management Console for the first time, you must change the password of your username. Your password must meet complexity requirements. The password must be 8 to 20 characters in length and must contain at least two of the following character types:
>
> - Uppercase or lowercase letters
> - Digits
> - Special characters, which include ! @ # $ %

3. Click **Login**.
4. In the top navigation bar, choose **Products > Application Services > API Gateway**.

1.3. Quick start

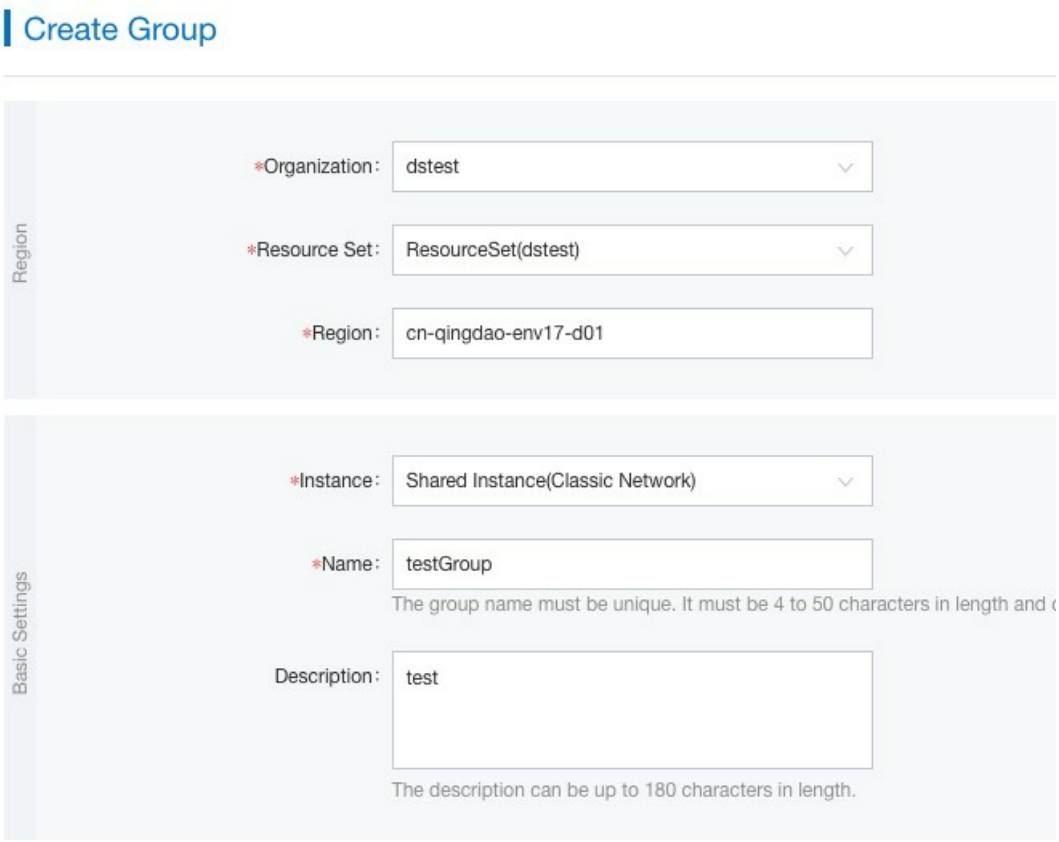
1.3.1. Create an API with HTTP as the backend service

This topic describes how to create and publish an API with HTTP as the backend service in API Gateway. This topic also describes how to call the API by using the AppKey and AppSecret pair of an app, with Security Certification set to Alibaba Cloud APP. You need to perform the following operations in sequence: **create an API group**, **create an API**, **create an app and an API authorization**, **debug the API**, and **call the API**.

Create an API group

APIs are managed in API groups. Before you create an API, you must create an API group.

1. **Create a group**: In the left-side navigation pane of the API Gateway console, choose **Publish APIs > API Groups**. On the Group List page, click **Create Group** in the upper-right corner. On the Create Group page, specify **Organization**, **Resource Set**, and **Region**, set Name to testAppkeyGroup, and then click **Submit**. After you specify Organization, the Instance parameter is automatically set to Shared Instance(Classic Network).

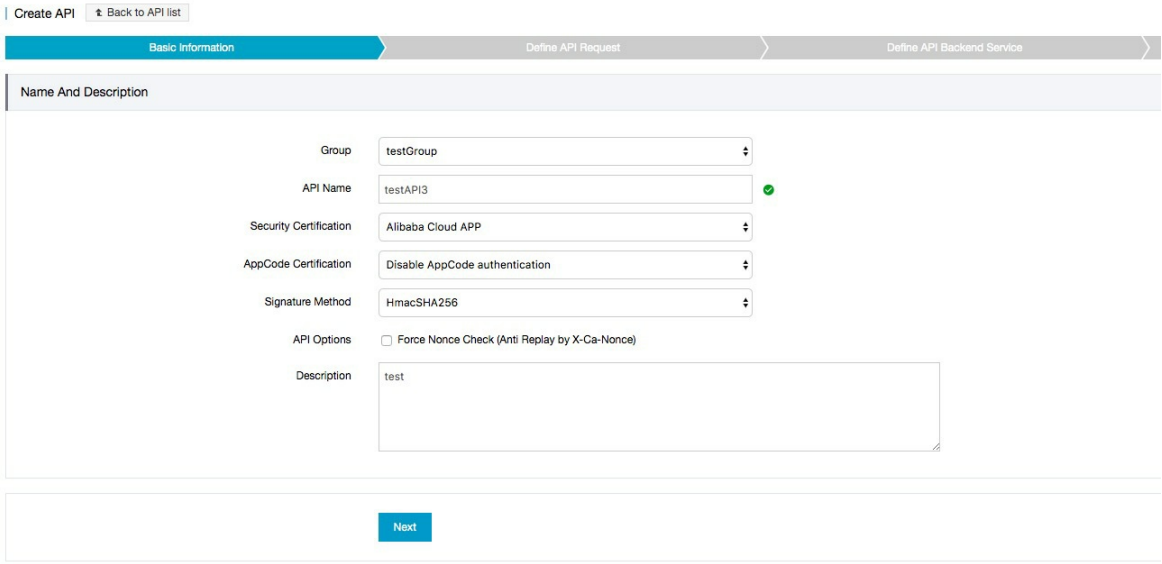


2. **View group information**: In the Submitted message, click Back to Console. On the **Group List** page, click **Refresh** in the upper-right corner. The group that you created is displayed. You can click the group name to go to the Group Details page and perform operations such as binding a domain name and modifying basic information. A second-level domain is automatically created for the API group. It can be used in Apsara Stack to call all APIs within this group. In this example, the domain name is used for tests.

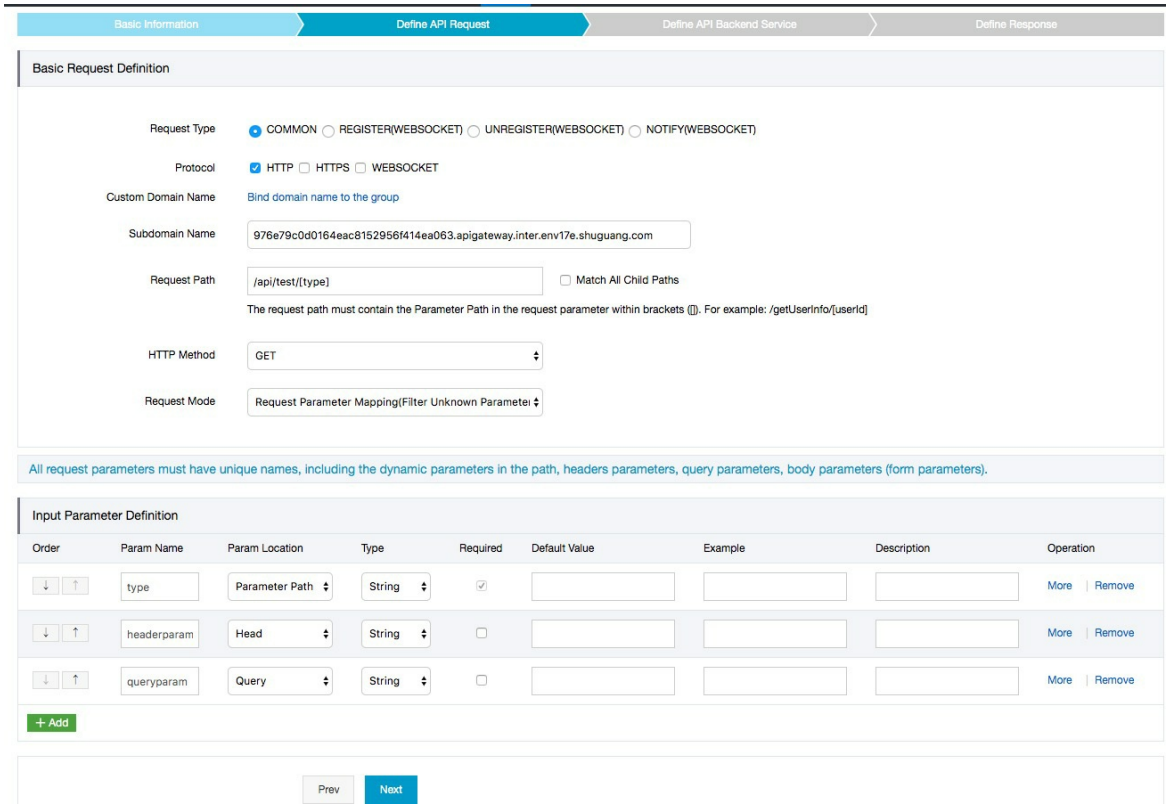
Create an API

In the left-side navigation pane of the API Gateway console, choose **Publish APIs > APIs**. On the API List page, click **Create API** in the upper-right corner. On the Create API page, perform the following steps:

1. **Specify basic information**. In this step, specify basic information, including Group, API Name, Security Certification, and Description. In this example, set Group to testAppkeyGroup, Security Certification to Alibaba Cloud APP, and AppCode Certification to Disable AppCode authentication, configure other parameters as required, and then click Next.

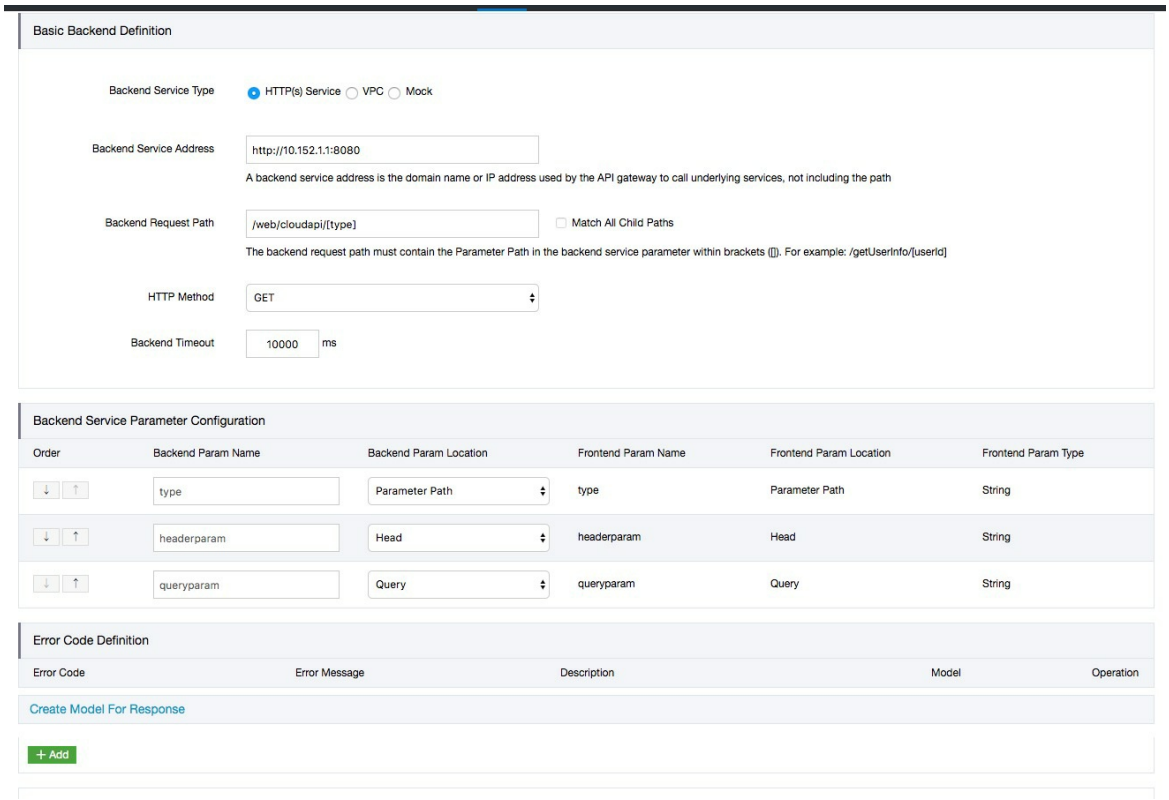


2. **Define an API request**. In this step, define how a client, such as a browser, a mobile app, or a business system, sends a request for the API. The parameters that need to be specified in this step include Request Type, Protocol, Request Path, HTTP Method, Request Mode, and those in the Input Parameter Definition section. Then, click Next. In this example, enter /web/cloudapi in the Request Path field and configure a path parameter, a query parameter, and a header parameter in the Input Parameter Definition section.

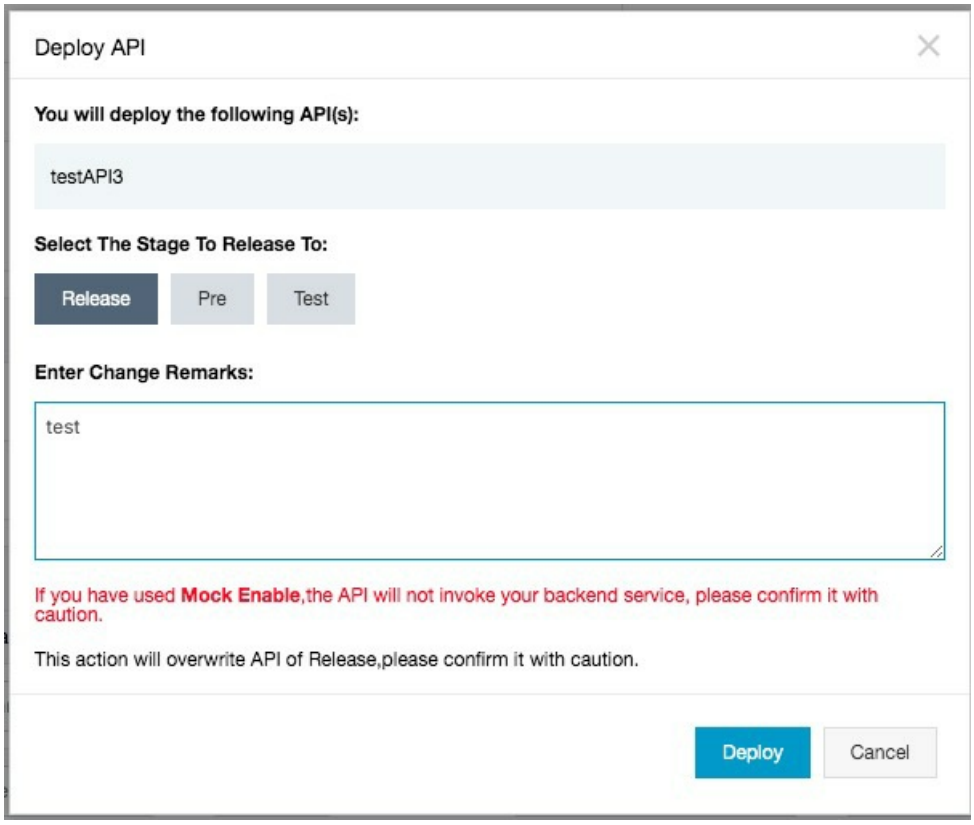


3. **Specify API backend service information**. In this step, configure a backend service type and a backend service address of the API and the mappings between request and response parameters. In this example, set Backend Service Type to HTTP(s) Service and Backend Service Address to an address that you can use to access API Gateway. For information about other backend service types, see API Gateway documentation. Configure other parameters such as Backend Request Path

as prompted and click Next.



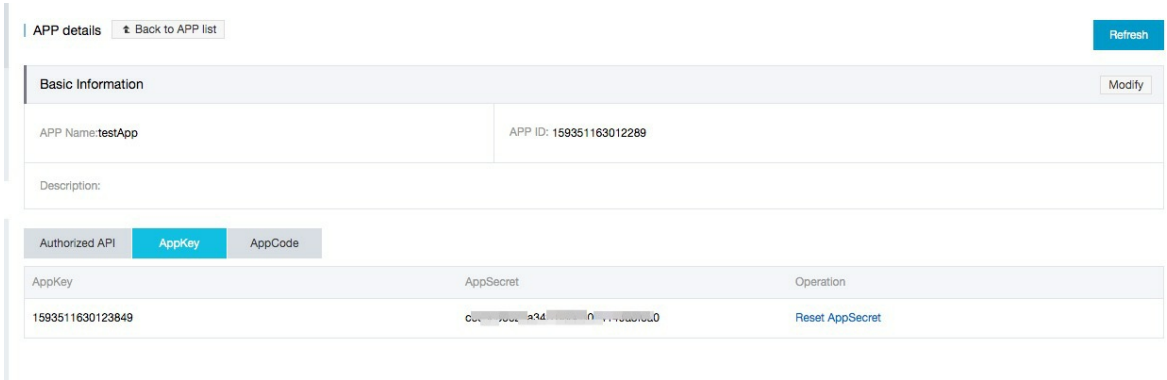
4. **Define return results**. In this step, configure response information to generate API documentation. The documentation helps API callers better understand APIs. You can configure parameters such as ContentType of Response, Sample of Returned Results, and Sample of Returned Failure. The configurations in this step are not involved in this example. Click **Create**.
5. **Publish the API**. API Gateway provides three environments to which you can publish an API: Release, Pre, and Test. All configurations you perform on an API can take effect only after you publish the API to a required environment. In this example, click Deploy in the message that indicates successful API creation. Alternatively, find the created API on the API List page and click Deploy in the Operation column. In the Deploy API dialog box, set Select The Stage To Release To to Release, specify Enter Change Remarks, and then click Deploy.



Create an app and an API authorization

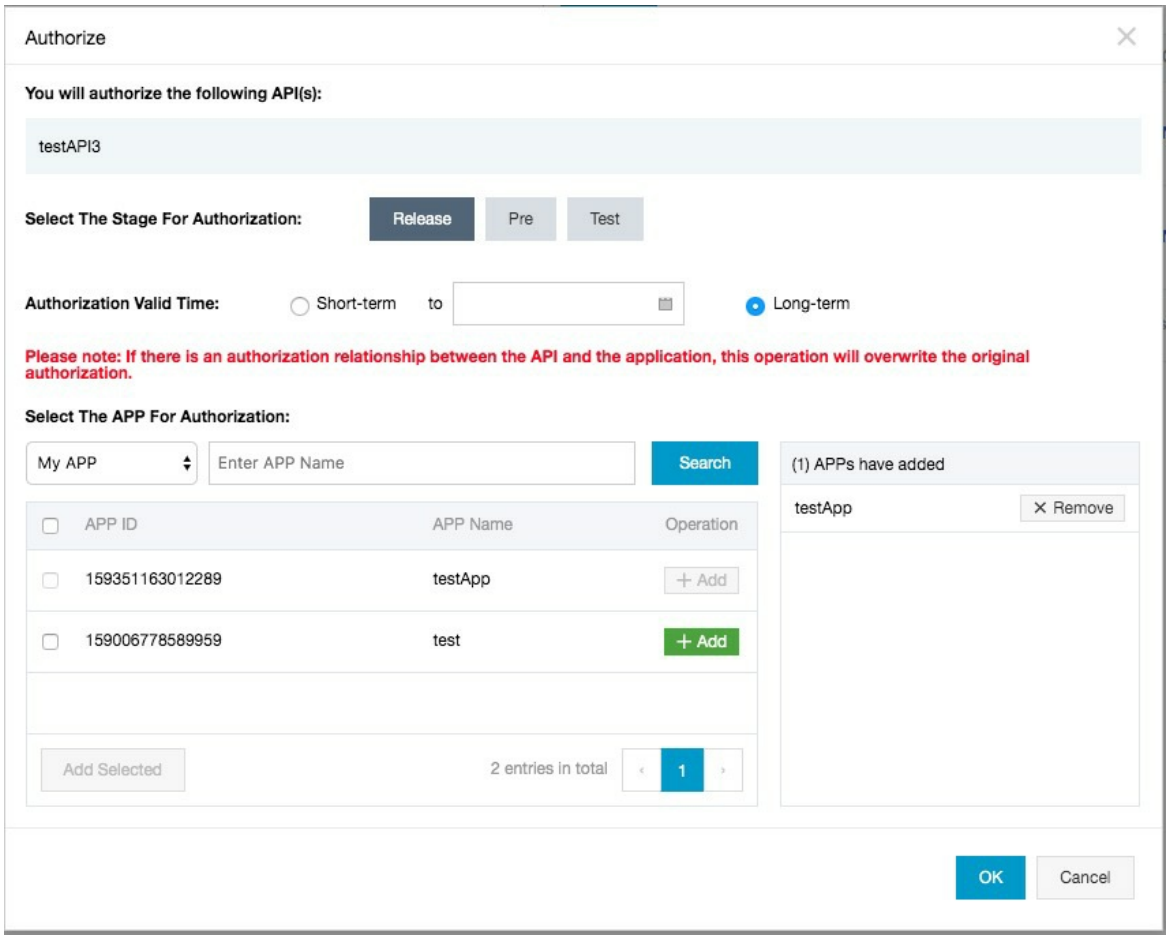
Apps are the identities that you use to call APIs. In Step 1 of the "Create an API" section, Security Certification is set to Alibaba Cloud APP. Therefore, after you publish the API, you must create an app and authorize the app to call the API.

1. **Create an app**: In the left-side navigation pane of the API Gateway console, choose **Consume APIs > APPs**. On the APP List page, click Create APP in the upper-right corner to create an app. Then, click the name of the created app to go to the APP details page, as shown in the following figure. Two authentication modes are provided: an AppKey and AppSecret pair and AppCode. Each app has an AppKey and AppSecret pair. It works in a way similar to an account and password pair. When you call an API, you must pass in the AppKey as an input parameter. AppSecret is used to calculate the signature string. API Gateway authenticates the key pair to verify your identity.



2. **Authorize an API**: On the **API List** page, find the created API and click **Authorize** in the Operation column. In the dialog box that appears, set Select The Stage For Authorization to the environment in which you published the API. In this example, set this parameter to Release. Click **Search** to search for the created app and click **+Add** in the Operation column to add this app to

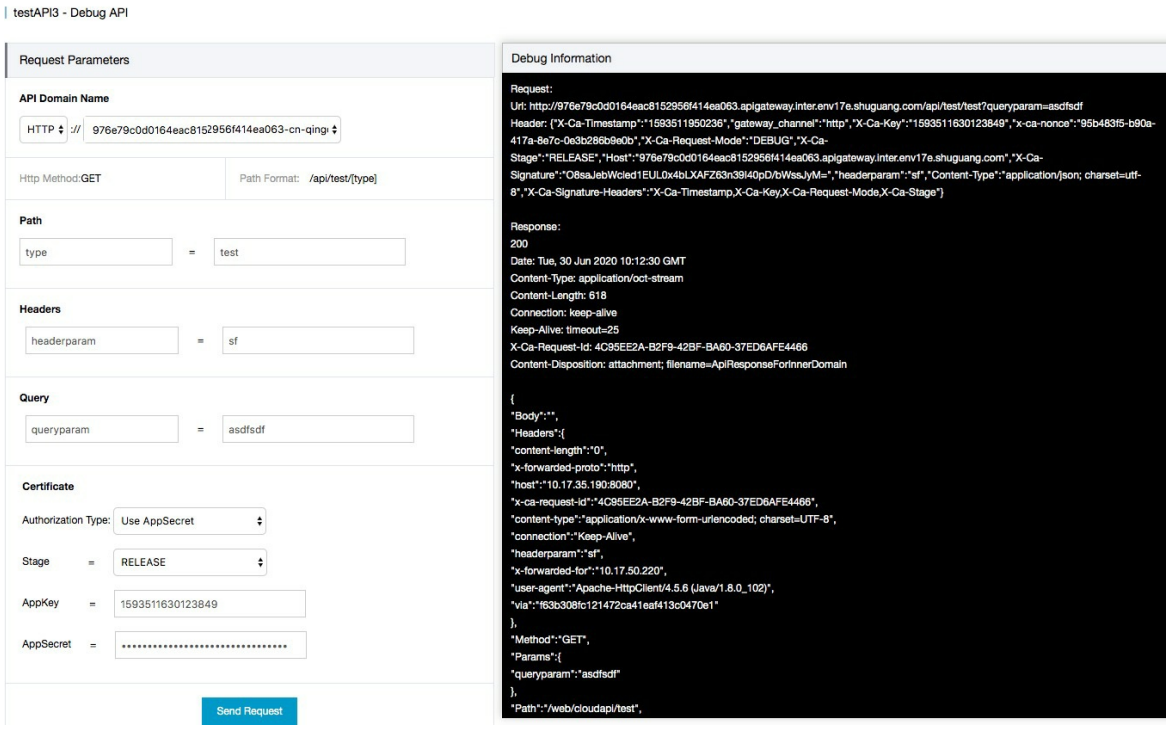
the pane on the right. Then, click **OK**. A success message appears.



Debug an API

API Gateway supports online debugging. We recommend that you use this feature to check whether an API is correctly configured before you call this API at the client side.

1. In the left-side navigation pane of the API Gateway console, choose **Consume APIs > APPs**. On the APP List page, click the name of the app that has been authorized to call the created API. On the APP details page, click **Authorized API**, find the desired API, and then click **Debug API** in the Operation column. On the API debugging page, if you have defined input parameters for this API, you can enter different values for the input parameters to check whether the API is correctly configured.



Only the APIs that are published to a required environment and can be called by authorized apps are displayed after you click **Authorized API**.

Call an API

After you debug and publish an API to a Release environment, you can use SDKs for API Gateway to call the API in your business system.

1. In the left-side navigation pane of the API Gateway console, choose **Consume APIs > Authorized APIs SDK**. On the Authorized APIs SDK Auto-Generation page, find the desired app and click the required programming language in the Authorized APIs SDK Auto-Generation column to download the related SDK package. The SDK package contains the API documentation and the SDK for the created API. For information about how to use the SDK, see the Readme file in the SDK package.

   Only the SDKs for APIs that are published to a Release environment are supported.

1.4. Call an API

1.4.1. Manage applications

1.4.1.1. Create an app

Apps are the identities that you use to call APIs. You can own multiple apps. Your apps can be authorized to call different APIs based on your business requirements. User accounts cannot be authorized to call APIs. In the API Gateway console, you can create, modify, or delete apps, view the details of apps, manage key pairs, and view the APIs that can be called by authorized apps.

Each app has an **AppKey** and **AppSecret** pair. It works in a way similar to an account and password pair. When you call an API, you must pass in the **AppKey** as an input parameter. **AppSecret** is used to calculate the signature string. API Gateway authenticates the key pair to verify your identity. An app must be authorized to call an API. Both authorization and authentication are intended for apps.

You can create apps on the **APP List** page in the API Gateway console.

Procedure

1. Log on to the API Gateway console.
2. In the left-side navigation pane, click **Consume APIs** and then **APPs**.
3. On the APP List page, click **Create APP** in the upper-right corner.

   The app name must be globally unique. It must be 4 to 26 characters in length and can contain only letters, digits, and underscores (_). It must start with a letter.

   After an app is created, the system automatically assigns an **AppKey** and **AppSecret** pair to the app. You must use the **AppSecret** to calculate a signature string. When you call an API, you must include the signature string in the request. API Gateway verifies your identity based on the signature string.

   On the **APP List** page, click the app name to go to the APP details page that displays the **AppKey** and **AppSecret** information. If the key pair is missing, you can reset it.

1.4.1.2. View app details

You can view details of created apps.

Procedure

1. Log on to the API Gateway console.
2. In the left-side navigation pane, click **Consume APIs** and then **APPs**.
3. On the APP List page, click the name of the app that you want to view.

   On the APP details page, you can view basic app information. You can also click **AppKey** or **Authorized API** to view key pair information and APIs that can be called by authorized apps.

1.4.1.3. Edit an app

You can edit a created app.

Procedure

1. Log on to the API Gateway console.
2. In the left-side navigation pane, click **Consume APIs** and then **APPs**.
3. On the APP List page, find the target app and click **Edit** in the Operation column.
4. In the Modify APP dialog box, modify app information and click **OK**.

1.4.1.4. Delete an app

You can delete a created app.

Procedure

1. Log on to the API Gateway console.
2. In the left-side navigation pane, click **Consume APIs** and then **APPs**.
3. On the APP List page, find the target app and click **Delete** in the Operation column.
4. In the Confirm Deletion message, click **OK**.

1.4.2. View created APIs

You can view created APIs in the API Gateway console.

Procedure

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > APIs**.

1.4.3. Authorize an application

Authorization is the process of authorizing an application to call an API. Your applications must be authorized before they can call APIs.

You must provide your application IDs to the API provider for authorization. After authorization, you can view the APIs that your applications have been authorized to call in the API Gateway console.

The APIs that your applications have been authorized to call are displayed in the **Callable APIs** section on the application details page.

After the API provider authorizes your applications to call APIs, you do not need to and cannot authorize your applications.

1.4.4. Encrypt a signature

When you call an API, you must construct a signature string and add the calculated signature string to the request header. API Gateway uses symmetric encryption to verify the identity of the request sender.

- Add the calculated signature string to the request header.
- Organize the request parameters into a string-to-sign based on Request signatures. Then, use the algorithm provided in the SDK sample to calculate the signature. The result is the calculated signature string.
- Both HTTP and HTTPS requests must be signed.

For more information about how to construct a string-to-sign, see Request signatures. Replace the AppKey and AppSecret in the SDK sample with your own AppKey and AppSecret. Then, construct a string-to-sign based on Request signatures. After creating the string-to-sign, you can use it to initiate a request.

1.4.5. Request signatures

Endpoint

- Each API belongs to an API group, and each API group has a unique endpoint. An endpoint is an independent domain name that is bound to an API group by the API provider. API Gateway uses endpoints to locate API groups.

- An endpoint must be in the *www.[Independent domain name].com/[Path]?[HTTPMethod]* format.
- API Gateway locates a unique API group by endpoint, and locates a unique API in the group through the combination of Path and HTTPMethod.
- After you purchase an API, you can obtain the API documentation from the **Purchased APIs** list in the API Gateway console. If you have not purchased an API, you must obtain authorization from the API provider for your applications to call the API. After authorization, you can obtain the API documentation from the **Callable APIs** list on the application details page.

System-level header parameters

- (Required) X-Ca-Key: AppKey.
- (Required) X-Ca-Signature: the signature string.
- (Optional) X-Ca-Timestamp: the timestamp passed in by the API caller. This value is a UNIX timestamp representing the number of milliseconds that have elapsed since January 1, 1970 00:00:00 UTC. The timestamp is valid for 15 minutes by default.
- (Optional) X-Ca-Nonce: the UUID generated by the API caller. To prevent replay attacks, you must specify both the X-Ca-Nonce header and the X-Ca-Timestamp header.
- (Optional) Content-MD5: When the request body is not a form, you can calculate the MD5 value of the request body. Then, you can send the value to API Gateway for MD5 verification.
- (Optional) X-Ca-Stage: the stage of the API. Valid values: TEST, PRE, and RELEASE. Default value: RELEASE. If the API that you intend to call has not been published to the release environment, you must specify the value of this parameter. Otherwise, a URL error will be reported.

Signature validation

Construct the signature calculation strings

```
String stringToSign=
HTTPMethod + "\n" +
Accept + "\n" +          // We recommend that you specify the Accept header in the request. If the request hea
der is not set, some HTTP clients will use the default value */*, causing signature verification to fail.
Content-MD5 + "\n"
Content-Type + "\n" +
Date + "\n" + ,
Headers +
Url
```

An HTTP method must be uppercase, such as POST.

If Accept, Content-MD5, Content-Type, and Date are empty, add a line break `\n` after each of them.

If Headers is empty, `\n` is not required.

Content-MD5

Content-MD5 indicates the MD5 value of the request body. The value is calculated as follows:

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

bodyStream indicates a byte array.

Headers

Headers indicates the string constructed by the keys and values of the header parameters that are used for Headers signature calculation. We recommend that you use the parameters starting with X-Ca and custom header parameters for signature calculation.

> **Notice**
>
> The following parameters are not used for Headers signature calculation: X-Ca-Signature, X-Ca-Signature-Headers, Accept, Content-MD5, Content-Type, and Date.

Headers construction method:

Sort the header keys used for Headers signature calculation **in alphabetical order**. Construct the string based on the following rules: If the value of a header parameter is empty, use `HeaderKey + ":" + "\n"` for signature calculation. The key and colon (:) must be retained.

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

The keys of the header parameters used for Headers signature calculation must be separated with commas (,), and placed in the request headers. The key is X-Ca-Signature-Headers.

Url

Url indicates the **Form** parameter in **Path + Query + Body**. For **Query + Form**, sort keys specified by **Key** in alphabetical order and construct the string based on the following rules: If **Query** or **Form** is empty, no question marks `?` are required for `Url = Path`. If **Value** of a parameter is empty, only **Key** is used for signature calculation and an equal sign (=) is not required.

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

> **Notice**
>
> Note: The **Query** parameter or the **Form** parameter may have multiple values specified by **Value**. If both parameters have multiple values, only the first value of each parameter is used for signature calculation.

Signature calculation

```
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` indicates the AppSecret.

Signature passing

Add the calculated signature to the request header. The key is X-Ca-Signature.

Signature troubleshooting

If signature verification fails, API Gateway places the returned stringToSign value in the HTTP response header and sends the response to the client. The key is X-Ca-Error-Message. Compare the stringToSign value calculated by the client with the one returned by the server.

If the stringToSign values from the client and server are the same, check the AppSecret used for signature calculation.

HTTP headers do not support line breaks. Line breaks in stringToSign values are filtered out. Ignore the line breaks when you make a comparison.

Signature demo

For detailed demo (Java) of signature calculation, please refer to the API Gateway console.

1.4.6. API call examples

You can edit an HTTP or HTTPS request to call an API. The API Gateway console provides API call examples of multiple programming languages for you to test the call.

Part 1: Request

Request URL

When you call an API over an internal network, the second-level domain of the API group to which this API belongs is used by default. To view a second-level domain, choose **Publish APIs > API Groups** in the left-side navigation pane of the API Gateway console. Click the name of the target group to go to the Group Details page. If this group is bound with an independent domain, you can use this independent domain to initiate an access request.

```
http://e710888d3ccb4638a723ff8d03837095-cn-qingdao.aliapi.com/demo/post
```

Request method

`POST`

Request body

```
FormParam1=FormParamValue1&FormParam2=FormParamValue2  //HTTP request body
```

Request header

```
Host: e710888d3ccb4638a723ff8d03837095-cn-qingdao.aliapi.com
Date: Mon, 22 Aug 2016 11:21:04 GMT
User-Agent: Apache-HttpClient/4.1.2 (java 1.6)
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
// The request body type. Set the request body type based on the actual request you want to make.
Accept: application/json
// The response body type. Some APIs can return data in the appropriate format based on the specified response type. We recommend that you manually specify the request header. If the request header is not specified, some HTTP clients will use the default value */*, which causes a signature error.
X-Ca-Request-Mode: debug
// Specifies whether to enable the debug mode. This parameter is not case-sensitive. If it is not specified, the debug mode is disabled. Enable this mode in the API debugging phase.
X-Ca-Version: 1
// The API version number. Currently, all APIs support only version 1. You can leave this request header unspecified. The default version number is 1.
X-Ca-Signature-Headers: X-Ca-Request-Mode,X-Ca-Version,X-Ca-Stage,X-Ca-Key,X-Ca-Timestamp
// The custom request headers involved in signature calculation. The server reads the request headers based on this configuration to sign the request. This configuration does not include the Content-Type, Accept, Content-MD5, and Date request headers, which are already included in the basic signature structure. For more information about the signature, see Request signatures.
X-Ca-Stage: RELEASE
// The stage of the API. Valid values: TEST, PRE, and RELEASE. This parameter is not case-sensitive. The API provider can select the stage to which the API is published. The API can be called only after it is published to the specified stage. Otherwise, the system will prompt that the API cannot be found or that the request URL is invalid.
X-Ca-Key: 60022326
// The AppKey of the request. You must obtain the AppKey in the API Gateway console. Apps can call APIs only after they have been authorized.
X-Ca-Timestamp: 1471864864235
// The request timestamp. This value is a UNIX timestamp that represents the number of milliseconds that have elapsed since January 1, 1970 00:00:00 UTC. The timestamp is valid for 15 minutes by default.
X-Ca-Nonce:b931bc77-645a-4299-b24b-f3669be577ac
// The unique ID of the request. AppKey, API, and Nonce must be unique within the last 15 minutes. To prevent replay attacks, you must specify both the X-Ca-Nonce header and the X-Ca-Timestamp header.
X-Ca-Signature: FJleSrCYPGCU7dMlLTG+UD3Bc5Elh3TV3CWHtSKh1Ys=
// The request signature.
CustomHeader: CustomHeaderValue
// The custom request headers. CustomHeaderValue is used as an example. You can configure multiple custom request headers in requests based on the definition of the API that is being called.
```

Part 2: Response

Status code

```
400  // The status code of the response. If the value is greater than or equal to 200 but less than 300, the call succeeded. If the value is greater than or equal to 400 but less than 500, a client-side error has occurred. If the value is greater than 500, a server-side error has occurred.
```

Response header

```
X-Ca-Request-Id: 7AD052CB-EE8B-4DFD-BBAF-EFB340E0A5AF
// The unique ID of the request. When API Gateway receives a request, it generates a request ID and returns t
he request ID to the client in the X-Ca-Request-Id header. We recommend that you record the request ID in b
oth the client and backend server for troubleshooting and tracking.
X-Ca-Error-Message: Invalid Url
// The error message returned by API Gateway. If a request fails, API Gateway returns the error message to th
e client in the X-Ca-Error-Message header.
X-Ca-Debug-Info: {"ServiceLatency":0,"TotalLatency":2}
// The message returned only when the debug mode is enabled. The message is used only for reference at th
e debugging stage.
```

Regardless of whether you call an API by using HTTP or HTTPS, the request must include the signature information. For information about how to calculate and deliver an encrypted signature, see Request signatures.

1.5. APIs

1.5.1. Manage groups

1.5.1.1. Create an API group

You can create an API group in the API Gateway console.

Procedure

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > API Groups**.
3. On the Group List page, click **Create Group** in the upper-right corner.
4. On the Create Group page, specify **Organization**, **Resource Set**, and **Region** in the Region section. Then specify **Name** and **Description** in the Basic Settings section and click **Submit**.

   The group name must be unique. It must be 4 to 50 characters in length and can contain only letters, digits, and underscores (_). It must start with a letter.

1.5.1.2. Manage domain names

In Apsara Stack, you can use the second-level domain of a group to directly call an API that belongs to this group. You can also bind your domain name to the group so that you can use your domain name to call APIs that belong to the group.

Context

If you want to use your domain name to directly call APIs that belong to a group, you must bind the domain name to the group and add a DNS record to your domain name. The domain name must be resolved to the second-level domain of the group or the IP address that corresponds to the second-level domain.

Bind an independent domain

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > API Groups**.

3. On the Group List page, find the target group and click **Bind Domain** in the **Operation** column.
4. In the Bind Domain Name dialog box, specify **Domain Name** and click **OK**.

Delete an independent domain

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > API Groups**.
3. On the **Group List** page, find the target group and click its name to go to the **Group Details** page.
4. In the **Custom Domain Name** section, click **Delete Domain** in the Operation column.
5. In the Confirm Deletion message, click **OK**.

1.5.1.3. Manage certificates

To use HTTPS on an independent domain, you must upload an SSL certificate.

Context

To perform HTTPS API calls, you must use a domain name that supports HTTPS and set Protocol to HTTPS in the Basic Request Definition section when you define an API request.

Upload a certificate

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > API Groups**.
3. On the **Group List** page, click the name of the target group to go to the **Group Details** page.
4. In the **Custom Domain Name** section, click **Create Certificate** in the **SSL Certificate** column.
5. In the **Create Certificate** dialog box, specify **Certificate Name**, **Certificate Content**, and **Private Key**, and click **OK**.

Delete a certificate

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > API Groups**.
3. On the **Group List** page, click the name of the target group to go to the **Group Details** page.
4. In the **Custom Domain Name** section, click **Delete Certificate** in the **Operation** column.
5. In the Confirm Deletion message, click **OK**.

1.5.1.4. Delete an API group

You can delete a created API group.

Procedure

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > API Groups**.
3. On the Group List page, find the target group and click **Delete** in the Operation column.
4. In the Delete Group message, click **Delete**.

> **Note** Before you delete a group, you must delete APIs that belong to this group.

1.5.1.5. Manage environments

To understand environment management, you must be familiar with two concepts: environment and environment variable.

- An **environment** is an API group configuration. You can configure several environments for a group. APIs that have not been published are considered defined APIs. An API can provide external services only after it is published to an environment.
- **Environment variables** are environment-specific variables that you can create and manage. For example, you can create an environment variable named `Path` in the Release environment. The value of this variable is `/stage/release`.

When you define an API, you can add variables, in the format of #Variable name#, to Path values. For example, specify `Path` in the format of `#Path#` when you define an API.

When you publish the API to the Release environment, the value of `#Path#` is `/stage/release`.

When you publish the API to another environment that does not have the environment variable `#Path#`, the variable value cannot be obtained and the API cannot be called.

Environment variables allow backend services to run in different runtime environments. You can access various backend services by configuring the same API definition but different backend service endpoints and paths across different environments. When you use environment variables, consider the following limits:

- Variable names are case-sensitive.
- If you configure a variable in the API definition, you must configure the name and value of the variable for the environment to which the API is published. Otherwise, no value is assigned to the variable and the API cannot be called.

Create an environment variable

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > API Groups**.
3. On the Group List page, find the desired group and click **View Stages** in the Operation column.
4. On the Stage Management page, click **Add Variable** in the upper-right corner. In the Add Variable dialog box, specify *Name* and *Value* and click **Add**.

> **Notice** The variable names for the Release, Pre, and Test environments must be the same. However, the variable values for the three environments can be different. After an API is published to a specified environment, the variable value will be automatically replaced.

Delete an environment variable

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > API Groups**.
3. On the Group List page, find the desired group and click **View Stages** in the Operation column.
4. On the Stage Management page, select the runtime environment, find the desired variable, and

then click **Delete** in the Operation column.

5. In the Confirm Deletion message, click **OK**.

1.5.2. Create an API

1.5.2.1. Overview

Creating an API is the process of defining the API in the API Gateway console. When creating an API, you must define the basic information, back-end service information, API request information, and response information of the API.

- API Gateway enables you to configure verification rules for input parameters. API Gateway can be configured to pre-verify and forward API requests that contain valid parameters.
- API Gateway enables you to configure mappings between front-end and back-end parameters. API Gateway can map a front-end parameter at one location to a back-end parameter at a different location. For example, you can configure API Gateway to map a `Query` parameter in an API request to a `Header` parameter in a back-end service request. In this way, you can encapsulate your back-end services into standard API operations.
- API Gateway enables you to configure constant and system parameters. These parameters are not visible to your users. API Gateway can add these parameters to requests based on your business requirements before sending the requests to your back-end services. If you want API Gateway to attach the keyword `apigateway` to each request that API Gateway forwards to your back-end services, you can configure `apigateway` as a constant parameter and specify where it is received.

1.5.2.2. Create an API

Procedure

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > APIs**.
3. On the API List page, click **Create API** in the upper-right corner.
4. Specify basic information of the API and click **Next**.

Parameter	Description
Group	The basic management unit of APIs. Before you create an API, you must create an API group. When you select a group, a region is selected for the API.
API Name	The name of the API to be created.
Security Certification	The authentication mode of API requests. Valid values: *Alibaba Cloud APP* and *No Certification*. ○ *Alibaba Cloud APP*: This mode requires the requester to pass the app authentication to call an API. ○ *No Certification*: This mode allows all users who know the request definition of an API to initiate a request. API Gateway directly forwards the request to your backend service without the need to verify the identity of a requester.

Parameter	Description
Signature Method	The algorithm that is used to sign API requests. Valid values: ◦ HmacSHA256 ◦ HmacSHA1 and HmacSHA256: If you set this parameter to this value, both the algorithms are supported.
Description	The description of the API.

5. Define an API request. In this step, define how users call your API, with the following parameters specified: Request Type, Protocol, Request Path, HTTP Method, and Request Mode.

Parameter	Description
Request Type	The request type. Only the COMMON request type is supported. Valid values: COMMON, REGISTER(WEBSOCKET), UNREGISTER(WEBSOCKET), and NOTIFY(WEBSOCKET). ◦ *COMMON*: indicates common HTTP or HTTPS requests. ◦ *REGISTER(WEBSOCKET)*: indicates the bidirectional control signaling to register devices. It is sent from the client to the server. ◦ *UNREGISTER(WEBSOCKET)*: indicates the bidirectional control signaling to deregister devices. It is sent from the client to the server. After devices are deregistered, server-to-client notifications are no longer received. ◦ *NOTIFY(WEBSOCKET)*: After the backend service receives the registration signaling sent from the client, the backend service records the device ID and sends a server-to-client notification to API Gateway. Then, API Gateway sends the notification to the device. If the device is online, API Gateway sends the server-to-client notification to the device.
Protocol	The supported protocol. Valid values: *HTTP*, *HTTPS*, and *WEBSOCKET*.
Request Path	The API request path that corresponds to the service host. The request path can be different from the actual backend service path. You must specify a valid and semantically accurate path as the request path. You can configure dynamic parameters in the request path. This requires that you specify path parameters in the request. In addition, the path parameters can be mapped to query and header parameters that are received by the backend service.
HTTP Method	The HTTP request method. Valid values: *PUT*, *GET*, *POST*, *PATCH*, *DELETE*, and *HEAD*.

Parameter	Description
Request Mode	The request mode. Valid values: *Request Parameter Mapping(Filter Unknown Parameters)*, Request Parameter Mapping(Passthrough Unknown Parameters), and *Request Parameter Passthrough*. ○ Request Parameter Mapping(Filter Unknown Parameters): You must configure request and response data mappings for query, path, and body form parameters. API Gateway transparently passes only the configured parameters to the backend service. Other parameters are filtered out. ○ Request Parameter Mapping(Passthrough Unknown Parameters): API Gateway maps and verifies only configured request parameters and transparently passes unknown parameters in a request to the backend service. ○ Request Parameter Passthrough: You do not need to configure query and body form parameters, but must configure path parameters in the Input Parameter Definition section. All parameters sent from the client are transparently passed by API Gateway to the backend service.

6. Define request parameters.

   In this step, define the request parameters of your API. You can specify different request parameters for different parameter paths. You can select Head, Query, Body, or Parameter Path from the Param Location drop-down list. When you configure a dynamic path parameter, you must provide a description of this dynamic parameter in the Input Parameter Definition section. The following data types are supported: String, Int, and Boolean.

   - The names of all parameters must be unique.
   - You can use the shortcut keys in the Order column to adjust the parameter order.
   - To delete a parameter that is no longer required, you can click **Remove** in the Operation column that corresponds to the parameter.

7. Configure parameter verification rules.

   To configure verification rules of a parameter, you can click **More** in the Operation column that corresponds to the parameter. For example, you can specify Max Length and Enumeration. API Gateway pre-verifies requests based on the verification rules. Requests with invalid parameters are not sent to your backend service. This significantly reduces the workload on your backend service.

8. Configure the backend service and click **Next**.

   In this step, define mappings between request and response parameters, and specify the API configurations of your backend service. Backend service configurations include Backend Service Address, Backend Request Path, Backend Timeout, and configurations in the Backend Service Parameter Configuration, Constant Parameter, and System Parameter sections. After API Gateway receives a request, it converts the format of the request into the format that is required by your backend service based on the backend service configuration. Then, API Gateway forwards the request to your backend service.

   > **Note** You can configure the following parameters: dynamic path parameters, header parameters, query parameters, body parameters (non-binary), constant parameters, and system parameters. Each parameter name must be globally unique. For example, you cannot specify a header parameter and a query parameter that have the same name.

i. Specify related parameters in the Basic Backend Definition section.

Parameter	Description
Backend Service Type	■ HTTP(s) Service: This option is selected by default. It indicates that API Gateway accesses the backend service over HTTP or HTTPS. If API Gateway can directly communicate with the backend service, select this option. ■ VPC: If the backend service is deployed in a virtual private cloud (VPC), select this option. ■ Mock: If you want to simulate expected return results, select this option.
VPC ID	The ID of the VPC where your backend service is deployed. This parameter is required when Backend Service Type is set to VPC.
Backend Service Address	The host of the backend service. ■ If Backend Service Type is HTTP(s) Service, set this parameter to a domain name or a value in the `http(s)://host:port` format. ■ If Backend Service Type is VPC, set this parameter to a value in the `http://ip:port` format.
Backend Request Path	The actual request path of your API on your backend server. If you want to receive dynamic parameters in the backend path, you must specify the locations and names of the corresponding request parameters to declare parameter mappings.
HTTP Method	The HTTP request method. Valid values: *PUT*, *GET*, *POST*, *PATCH*, *DELETE*, and *HEAD*.
Backend Timeout	The response time for API Gateway to access the backend service after API Gateway receives an API request. The response time starts from the time when API Gateway sends an API request to the backend service and ends at the time when API Gateway receives a response returned by the backend service. The response time cannot exceed 30s. If API Gateway does not receive a response from the backend service within 30s, API Gateway stops accessing the backend service and returns an error message.

ii. Configure parameters in the Backend Service Parameter Configuration section.

API Gateway can set up mappings between request and response parameters, including name mappings and parameter location mappings. API Gateway can map a path, header, query, or body request parameter to a response parameter at a different location. This way, you can package your backend service into a standardized and professional API form. This part declares the mappings between request and response parameters.

> **Note** The request and response parameters must be globally unique.

iii. Configure constant parameters in the Constant Parameter section.

If you want API Gateway to attach the apigateway tag to each request that API Gateway forwards to your backend service, you can configure this tag as a constant parameter. Constant parameters are not visible to your users. After API Gateway receives requests, it automatically adds constant parameters to the specified locations and then forwards the requests to your backend service.

iv. Configure system parameters in the System Parameter section.

By default, API Gateway does not send its system parameters to your backend service. If you require the system parameters, you can configure the related locations and names. The following table lists the system parameters.

Parameter	Description
CaClientIp	The IP address of the client that sends a request.
CaDomain	The domain name from which a request is sent.
CaRequestHandleTime	The time when a request is sent. It must be in GMT.
CaAppId	The ID of the app that sends a request.
CaRequestId	The unique ID of the request.
CaApiName	The name of the API.
CaHttpSchema	The protocol that is used to call an API. The protocol can be HTTP or HTTPS.
CaProxy	The proxy (AliCloudApiGateway).

9. Define responses and click **Create**.

In this step, specify ContentType of Response, Sample of Returned Results, and Sample of Returned Failure, and add configurations in the Error Code Definition section. API Gateway does not parse responses, but forwards the responses to API requesters.

1.5.2.3. Security authentication

The security authentication methods that are supported by API Gateway include Alibaba Cloud applications and none.

- Alibaba cloud applications: An application must be authorized by the API provider to call an API. An API caller must provide an AppKey and encrypted signature. Otherwise, the API request validation will fail. For more information about the signature method, see Encrypt a signature.
- None: The API can be called without authorization after it is published. The AppKey and encrypted signature are not required when you make an API request.

1.5.2.4. Configure a network protocol

HTTPS domain names are not supported in the API Gateway console. To use an HTTPS domain name, you can call the API operations of API Gateway.

To configure a network protocol, perform the following operations: Find the target API on the API List page in the API Gateway console, and click Manage in the Operation column. On the **API Definition** page, click **Edit** in the upper-right corner. On the page that appears, specify Protocol in the **Define API Request** step.

Valid values of Protocol:

- HTTP
- HTTPS
- WEBSOCKET

1.5.2.5. Configure a request body

You can configure a request body when the HTTP method is POST, PUT, or PATCH. You can use the following methods to configure the request body. The methods are mutually exclusive.

- Form-based request body: Add a request parameter in the Input Parameter Definition section of the Define API Request step on the Create API page, and select Body from the Param Location drop-down list. The configured request body can only be used to transmit form data.
- Non-form-based request body: If the body content to be transmitted is in the JSON or XML format, select Non-Form data, such as JSON, Binary data in the Request Body section of the Define API Request step on the Create API page. The size of a request body cannot exceed 8 MB.

1.5.2.6. Configure an API in Mock mode

In most cases, business partners can work in combination to develop a project. The project development process is hindered due to the interdependence among business partners. Misunderstandings can also arise and affect the development progress or even cause severe delays to the project. The Mock mode is used to simulate the predetermined API responses in the project development process. This reduces misunderstandings and improves development efficiency.

API Gateway provides a simple configuration process of an API in Mock mode.

Configure an API in Mock mode

Log on to the API Gateway console. In the left-side navigation pane, choose **Publish APIs > APIs**. On the API List page, find the target API and click **Manage** in the Operation column. On the API Definition page, click **Edit** in the upper-right corner.

On the page that appears, configure the Mock mode in the **Define API Backend Service** step.

1. Set **Backend Service Type** to **Mock**.
2. Specify Mock Result in the Mock Configuration section.

   Enter your responses as the Mock-based response body. Responses can be in the JSON, XML, or text format. Example:

```
{
"result": {
  "title": " Mock test for API Gateway",
}
}
```

Save the settings and then publish the API to the Test or Release environment for testing.

3. Specify **HTTP Status Code** based on HTTP status code specifications. Enter 200 to indicate a successful API request.
4. Specify **Mock Header**. You can click +Add Item to add a Mock response header based on your business requirements.

1.5.2.7. Return the Content-Type header

The value of the Content-Type header is only used to generate API documentation. It does not affect responses returned by the back-end service. The Content-Type header is returned by the back-end service.

1.5.3. API management

1.5.3.1. View and modify an API

You can view and modify an API based on your business requirements.

> **Note** If you modify an API that is published, the modifications are not immediately applied. You must republish the modified API to synchronize the changes to the Release environment.

Procedure

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > APIs**.
3. On the API List page, find the desired API.
   - Click **Manage** in the Operation column. On the API Definition page, you can view the information of the API.
   - Click **Edit** in the upper-right corner to edit the API based on your business requirements.

     The procedure to create an API is similar to that to modify an API. For more information about how to create an API, see Create an API. If you want to cancel modifications before they are submitted, click Cancel Edit in the upper-right corner of the edit page.

1.5.3.2. Publish an API

After you create an API, you must publish the API to the Test, Pre, or Release environment before it can be called.

- When you use a second-level or independent domain to access an API that is published to a specified environment, you must specify the environment in the request header.
- If you publish an API that already has a running version in the Test or Release environment, the running version is automatically overwritten by the new version within 15s. However, all historical versions and definitions are recorded. This allows you to roll the API back to an earlier version.
- You can unpublish an API in the Test or Release environment. The plug-in binding relationship or the app authorization relationship is retained after you unpublish an API. These relationships take effect again if the API is republished. You can also perform related operations to remove the authorization or unbind a required plug-in.

Step 1: Publish an API

After you create an API, you can publish the API to the Test environment to test the API first.

API Gateway allows you to manage different versions of APIs in the Test or Release environment. You can publish or unpublish the API, and switch the version of the API. The version switch takes effect in real time.

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > APIs**.
3. On the API List page, find the desired API and click **Deploy** in the Operation column.
4. In the Deploy API dialog box, specify Enter Change Remarks and click **Deploy**.

Step 2: Test the API

To simulate API requests, you can create an app and authorize the app to call your API.

You can compile code based on actual scenarios, or use the SDK samples provided by API Gateway to call your API.

You can publish the API to the Test or Release environment. If no independent domain is mapped to the group to which the API belongs, you can test or call the API by using a second-level domain. When you make an API request, set the X-Ca-Stage header to TEST, PRE, or RELEASE to specify the environment of the API. If you do not specify the header, the API will be invoked to the Release environment.

1.5.3.3. Authorize an app

You must authorize an app before it can call an API. After you publish an API to the Release environment, you must authorize apps to call the API. You can grant or revoke the authorization of an app to call an API. API Gateway verifies the authorization relationship.

> **Note**
> - You can authorize one or more apps to call one or more APIs.
> - If an API is published to both the Test and Release environments and an app is authorized to call the API in the Test environment, the app can call only the API in the Test environment.
> - You can find an app based on its ID.
> - If you want to revoke the authorization of an app to call an API, go to the Authorization page of the API. Then select the required app and click Revoke Authorization in the lower-left corner.

An app indicates the identity of a requester. Before testing or calling an API, you or your users must create an app that is used as the identity of a requester. Then, you must authorize the app to call the API.

> **Note** Authorizations are environment-specific. If you want to use an app to call an API in both the Test and Release environments, you must authorize the app in both environments. Otherwise, errors may occur due to the inconsistency between the authorized environment and the requested environment.

Procedure

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > APIs**.
3. On the API List page, find the target API and click **Authorize** in the Operation column.
4. In the Authorize dialog box, specify Select The Stage For Authorization and Select The APP For Authorization.

   My APP is automatically selected from the drop-down list on the left. Click **Search**. Apps created under your account appear.

   If you want to authorize an app created under a different account, select **APP ID** from the drop-down list on the left, enter the app ID in the search bar, and click **Search**.

   To view the ID of an app, click **Consume APIs** and then **APPs** in the left-side navigation pane. On the APP List page, click the name of the target app to go to the APP details page.
5. Select an app to be authorized and click +Add in the Operation column to add this app to the right pane. Alternatively, you can select multiple apps to be authorized at a time and click Add Selected in the lower-left corner of the page to add these apps to the right pane.
6. Click **OK** to complete the authorization.
7. Click Manage in the Operation column that corresponds to the target API. On the API Definition page, click **Authorization** in the left-side navigation pane to view the authorized apps.

1.5.3.4. Revoke an authorization

You can revoke the authorization of an app to call an API.

Procedure

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > APIs**.
3. On the API List page, click the name of the desired API for which you want to revoke the authorization. On the API Definition page, click **Authorization** in the left-side navigation pane.
4. Select desired apps and click **Revoke Authorization** in the lower-left corner.
5. In the Confirm authorization revocation message, click **OK**.

1.5.3.5. Unpublish an API

You can unpublish an API.

You can unpublish an API in the Test or Release environment. The binding or authorization relationships of policies, keys, and apps are retained after you unpublish an API. These relationships will take effect again if the API is republished. For more information about how to remove these relationships, see Revoke an authorization.

Procedure

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > APIs**.
3. On the API List page, find the desired API and click **Undeploy** in the Operation column.
4. In the Undeploy API message, click **Undeploy**.

1.5.3.6. View the version history of an API

You can view the version history of an API, including the version number, description, environment, publish time, and specific definition of each version.

Procedure

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > APIs**.
3. On the API List page, find the target API and click **Manage** in the Operation column to go to the API Definition page.
4. Click **Deployment History** in the left-side navigation pane. You can view the version history of this API.
5. On the Deployment History page, find the target version and click **View** in the Operation column.

1.5.3.7. Change the version of an API

When you view the version history of an API, you can select a different version to switch the API to this version. The selected version then replaces the previous version and takes effect in the specified environment.

Procedure

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > APIs**.
3. On the API List page, find the target API and click **Manage** in the Operation column to go to the API Definition page.
4. Click **Deployment History** in the left-side navigation pane.
5. Find the target version and click **Switch to this version** in the Operation column.
6. In the API Version Switch dialog box, enter the description and click **Switch**.

1.5.4. Plugin management

1.5.4.1. Use parameters and conditional expressions

In an `access control plugin`, `throttling plugin`, `backend routing plugin`, or `error code mapping plugin`, you can obtain parameters from requests, responses, and system context. Then, you can use conditional expressions to evaluate these parameters. This topic describes how to define parameters and write conditional expressions.

1. Define parameters

1. Definition method

   Before you use a conditional expression, you must explicitly define all the parameters required in this conditional expression in the parameters field. Example:

```
---
parameters:
 method: "Method"
 appId: "System:CaAppId"
 action: "Query:action"
 userId: "Token:UserId"
```

The parameters specified in `parameters` are key-value pairs of the string type.

- `key` indicates the name of a variable to be used in a conditional expression. The name must be unique and must conform to the following regular expression: `[a-zA-Z_][a-zA-Z0-9]+`.
- `value` indicates the location of a parameter. It is specified in the `{location}` or `{location}:{name}` format.
  - `location` indicates the location of a parameter. For more information, see the following table.
  - `name` indicates the name of a parameter, which is used to locate the parameter at a specific location. For example, `Query:q1` indicates the first value of the query string named q1.

2. Parameter locations

   Before you use a conditional expression, you must define the parameters that are required in this conditional expression. The following table describes parameters at specific locations that can be used by various plugins.

Location	Included in	Description
Method	Request	The HTTP request method, in uppercase, such as GET or POST.
Path	Request	The complete HTTP request path, such as `/path/to/query`.
StatusCode	Response	The HTTP status code in a backend response, such as 200 or 400.
ErrorCode	Response	Error codes.
Header	Request/Response	Use `Header:{Name}` to obtain the first value of the HTTP header that is specified by `{Name}`.
Query	Request	Use `Query:{Name}` to obtain the first value of the query string that is specified by `{Name}`.
Form	Request	Use `Form:{Name}` to obtain the first value of the form that is specified by `{Name}`.

Location	Included in	Description
Host	Request	Use `Host:{Name}` to obtain the template parameters of the matched wildcard domain names.
Parameter	Request	Use `Parameter:{Name}` to obtain the first value of the custom API parameter that is specified by {Name}.
BodyJsonField	Response	Use `BodyJsonField:{JPath}` to obtain the JSON string in the body of an API request or a backend response in JSONPath mode.
System	Request/Response	Use `System:{Name}` to obtain the value of the system parameter that is specified by `{Name}`.
Token	Request/Response	If JWT is used for authentication, use `Token:{Name}` to obtain the value of the parameter that is specified by `{Name}` in a token.

Rules for use:

- You can use the following plugins at the request phase: `access control plugin`, `throttling plugin`, and `backend routing plugin`. These plugins support only the parameters at the following locations: `Method`, `Path`, `Header`, `Query`, `Form`, `Parameter`, `System`, and `Token`.
- You can also use the `error code mapping plugin` at the response phase. This plugin supports only the parameters at the following locations: `StatusCode`, `ErrorCode`, `Header`, `BodyJsonField`, `System`, and `Token`.
- Parameters at the `Method`, `Path`, `StatusCode`, and `ErrorCode` locations are defined in the {location} format.
- If you use parameters at the `Header` location in a plugin at the request phase, headers from client requests are read. If you use these parameters at the response phase, headers from backend responses are read.
- Parameters at the `Parameter` location are available only for plugins at the request phase. A `frontend parameter`, instead of a `backend parameter`, is used to search for the parameter with the same name in the API definition. If no parameter with the same name exists, a null value is returned.
- A complete request path is returned from `Path`. If you require a parameter at the Path location, use the corresponding parameter at the `Parameter` location.
- Parameters at the `BodyJsonField` location are available only for the `error code mapping plugin`. Obtain the JSON string in the body of a backend response in `JSONPath` mode. For more

information, see Usage notes of JSONPath.

- If `JWT` is used for authentication, use `Token:{CliamName}` to obtain the value of the parameter specified by `{CliamName}` in a token. For more information, see the plugin documentation.

3. Usage notes of JSONPath

   JSONPath is available only for the `error code mapping` plugin at the `BodyJsonField` location. It is used to extract the `JSON` string in the body of a backend response. For more information about JSONPath, see the JSONPath overview documentation.

   Example: When you use the expression `code:"BodyJsonField:$.result_code"`, you can obtain the value of `result_code` from the following body. `code:ok` is parsed from the following body.

   ```
   { "result_code": "ok", "message": ... }
   ```

4. System parameters

Parameter	Description	Value
CaClientIp	The IP address of the request client.	Example value: 37.78.3.3.
CaDomain	The full domain name in a request, with a Host header.	Example value: api.foo.com.
CaAppId	The ID of the application that sends the request.	Example value: 49382332.
CaAppKey	The key of the application that sends the request.	Example value: 12983883923.
CaRequestId	The unique ID of the request generated by API Gateway.	Example value: CCE4DEE6-26EF-46CB-B5EB-327A9FE20ED1.
CaApiName	The API name.	Example value: TestAPI.
CaHttpSchema	The protocol used by the client to call operations.	Valid values: http, https, and ws.
CaClientUa	The UserAgent header of the client.	Used to transparently pass values uploaded by the client.
CaStage	The running environment of API Gateway.	Valid values: TEST, PRE, and RELEASE.

2. Write conditional expressions

You can use conditional expressions in plugins or other scenarios to evaluate parameters in a wide variety of scenarios.

1. Basic syntax
   - Conditional expressions are similar to SQL statements. Example: `$A > 100 and '$B = 'B'`.
   - An expression is in the following format: `{Parameter}{Operator}{Parameter}`. In the preceding example, you can specify a `variable` or a `constant` for `$A > 100`.

- A `variable` starts with $ and references a parameter defined in the context. For example, `q1:"Query:q1"` is defined in `parameters`. You can use the variable `$q1` in your expression. The value of this variable is the value of the `q1` query parameter in the request.
- A `constant` can be a `string`, `number`, or `Boolean value`. Examples: `"Hello",' foo', 100, -1, 0.1, and true`. For more information, see Value types and evaluation rules.
- The following `operators` are supported:
  - `=` and `==`: equal to.
  - `<>` and `!=`: not equal to.
  - `>`, `>=`, `<`, and `<=`: comparison.
  - `like` and `!like`: check whether a specific string matches a specified pattern. The percent sign `%` is used as a wildcard in the evaluation. Example: `$Query like 'Prefix%'`.
  - `in_cidr` and `!in_cidr`: specify the mask of an IP address. Example: `$ClientIp in_cidr '47.89.0.0/24'`.
- You can use `null` to check whether a parameter is empty. Example: `$A == null` or `$A != null`.
- You can use the operators `and`, `or`, and `xor` to combine different expressions in a right-to-left order by default.
- You can use parentheses `()` to specify the priority of conditional expressions.
- You can use `!()` to perform the logical negation operation on the enclosed expression. For example, the result of `!(1=1)` is false.
- The following built-in functions are used for evaluation in some special scenarios:
  - `Random()`: generates a parameter of the floating-point number type. The parameter value ranges from 0 to 1. This parameter is used in scenarios where random input is required, such as blue-green release.
  - `Timestamp()`: returns a UNIX timestamp representing the number of milliseconds that have elapsed since the epoch time January 1, 1970, 00:00:00 UTC.
  - `TimeOfDay()`: returns the number of milliseconds from the current time to 00:00 of the current day in GMT.

2. Value types and evaluation rules
   - The following value types are supported in expressions:
     - `STRING`: The value can be a string. Single quotation marks (' ') or double quotation marks (" ") can be used to enclose a string. Examples: `"Hello"` and `'Hello'`.
     - `NUMBER`: The value can be an integer or a floating-point number. Examples: `1001`, `-1`, `0.1`, and `-100.0`.
     - `BOOLEAN`: The value can be a Boolean value. Valid values: `true` and `false`.
   - For the operator types `equal to`, `not equal to`, and `comparison`, the following evaluation rules apply:
     - `STRING` type: uses the string order for evaluation. Examples:
       - `'123' > '10000'`: The result is true.
       - `'A123' > 'A120'`: The result is true.
       - `'' <'a'`: The result is true.

- `NUMBER` type: uses numerical values for evaluation. Examples:
  - `123 > 1000` : The result is false.
  - `100.0 == 100` : The result is true.
- `BOOLEAN` type: For Boolean values, true is greater than false. Examples:
  - `true == true` : The result is true.
  - `false == false` : The result is true.
  - `true > false` : The result is true.

- For the operator types `equal to` , `not equal to` , and `comparison` , if the value types before and after an operator are different, the following evaluation rules apply:
  - Assume that a value before an operator is of the `STRING` type and that after the operator is of the `NUMBER` type. If the value type before the operator can be changed to `NUMBER` , use numerical values for evaluation. Otherwise, use the string order for evaluation. Examples:
    - `'100' == 100.0` : The result is true.
    - `'-100' > 0` : The result is false.
  - Assume that a value before an operator is of the `STRING` type and that after the operator is of the `BOOLEAN` type. If the value type before the operator can be changed to `BOOLEAN` and the value is not case-sensitive, use `BOOLEAN` values for evaluation. Otherwise, except for the evaluation result of `!=` , all the other evaluation results are `false` . Examples:
    - `'True' == true` : The result is true.
    - `'False' == false` : The result is true.
    - `'bad' == false` : The result is false.
    - `'bad' != false` : The result is true. If the value before the operator is not `true` or `false` , only the result for `!=` is true.
    - `'bad' != true` : The result is true.
    - `'0' > false` : The result is false.
    - `'0' <= false` : The result is false.
  - Assume that a value before an operator is of the `NUMBER` type and that after the operator is of the `BOOLEAN` type. The result is false.
- The `null` value is used to check whether a parameter is empty. For the operator types `equal to` , `not equal to` , and `comparison` , the following evaluation rules apply:
  - If the `$A` parameter is empty, the result of `$A == null` is true, and the result of `$A != null` is false.
  - If the empty string `''` is not equal to `null` , the result of `'' == null` is false, and the result of `'' == ''` is true.
  - For the `comparison` operator type, if the value on either side of the operator is `null` , the result is false.
- `like` and `!like` operators are used to match the prefix, suffix, and inclusion of a string. The following evaluation rules apply:
  - In an expression, the value after the operator must be a constant of the `STRING` type. Example: `$Path like '/users/%'` .

- The `'%'` wildcard character in the value after the operator is used to match the prefix, suffix, or inclusion of a string. Examples:
  - Prefix matching: `$Path like '/users/%'` and `$Path !like '/admin/%'`
  - Suffix matching: `$q1 like '%search'` and `$q1 !like '%.do'`
  - Inclusion relation matching: `$ErrorCode like '%400%'` and `$ErrorCode !like '%200%'`
- If the value type before an operator is not `NUMBER` or `BOOLEAN`, change the type to `STRING` and then perform the evaluation.
- If the value before an operator is `null`, the result is `false`.

- `in_cidr` and `!in_cidr` operators are used to identify the mask of a CIDR block. The following evaluation rules apply:
  - The value after an operator must be a constant of the `STRING` type and must be an IPv4 or IPv6 CIDR block. Examples:
    - `$ClientIP in_cidr '10.0.0.0/8'`
    - `$ClientIP !in_cidr '0:0:0:0:0:FFFF::/96'`
  - If the value type before an operator is `STRING`, the value is considered an IPv4 CIDR block for evaluation.
  - If the value type before an operator is `NUMBER` or `BOOLEAN` or the value is empty, the result is `false`.
  - The `System:CaClientIp` parameter specifies the IP address of the client, which is used for evaluation.

3. Use cases

- The following expression indicates that the probability is less than 5%:

```
Random() < 0.05
```

- The following expression indicates that the requested API is published to the Test environment:

```
parameters:
  stage: "System:CaStage"
```

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

- The following expression indicates that the custom parameter UserName is set to Admin and the source IP address is `47.47.74.0/24`:

```
parameters:
 UserName: "Token:UserName"
 ClientIp: "System:CaClientIp"
```

```
$UserName = 'Admin' and $CaClientIp in_cidr '47.47.74.0/24'
```

- The following expression indicates that the AppId parameter is set to 1001, 1098, or 2011, and the protocol that is used by the API request is HTTPS:

```
pameters:
 CaAppId: "System:CaAppId"
 HttpSchema: "System:CaHttpSchema"
```

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

- The following expression indicates that the JSON string in a body contains result_code that is not ok when StatusCode in a response is 200:

```
parameters:
 StatusCode: "StatusCode"
 ResultCode: "BodyJsonField:$.result_code"
```

```
$StatusCode = 200 and ($ResultCode <> null and $ResultCode <> 'ok')
```

4. Limits

- A maximum of 16 parameters can be specified in a plugin.
- A conditional expression can contain a maximum of 512 characters.
- The size of a request or response body specified by BodyJsonField cannot exceed 16 KB. Otherwise, the settings will not take effect.

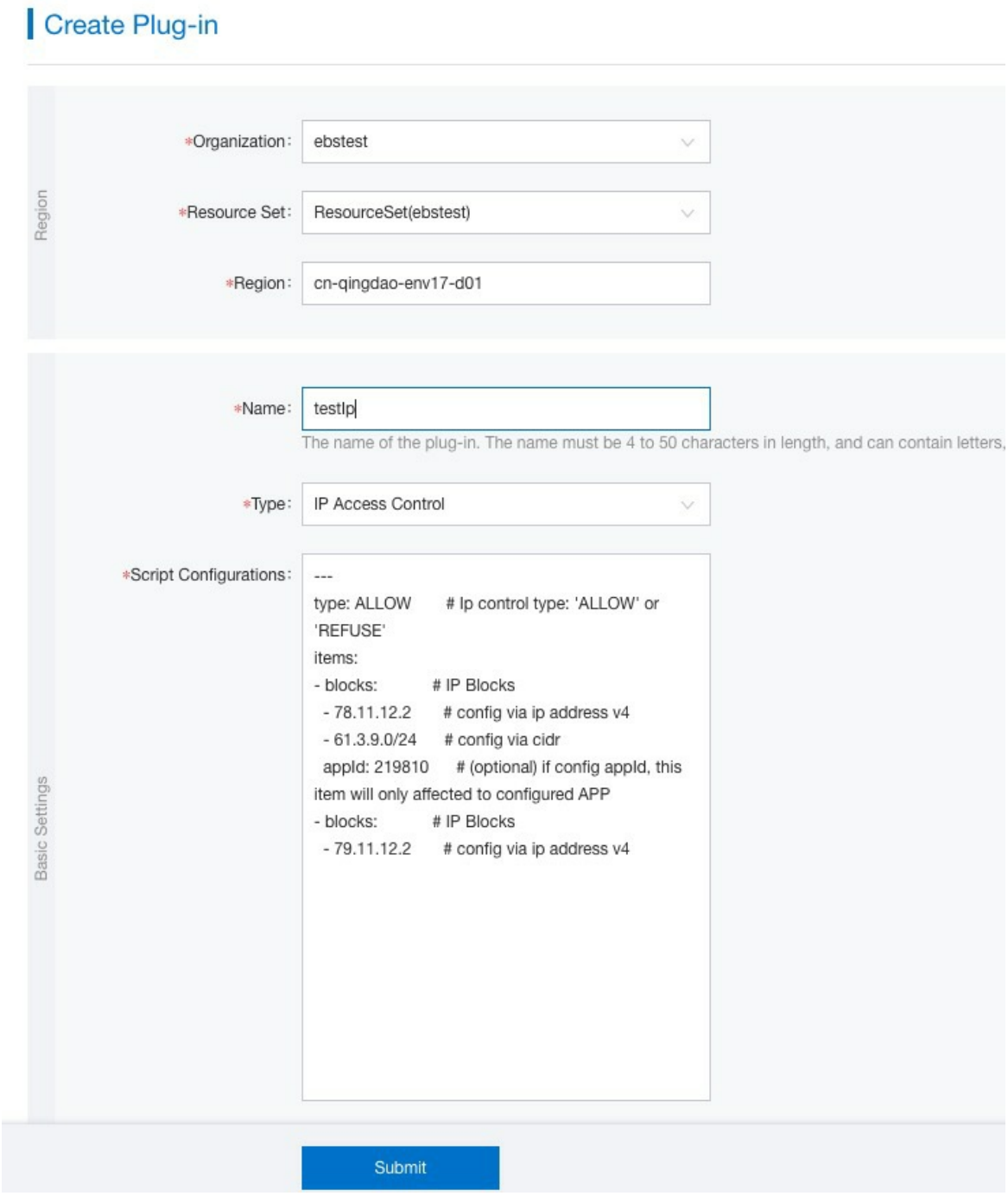
1.5.4.2. Create a plugin

1.5.4.2.1. Create an IP address-based access control plug-in

IP address-based access control helps API providers configure an IP address whitelist or blacklist for API calls. This topic describes how to create an IP address-based access control plug-in.

Procedure

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > Plugin**.
3. On the Plugins list page, click **Create Plugin** in the upper-right corner. On the Create Plug-in page, specify **Organization**, **Resource Set**, **Region**, and **Plug-in Name**, and set Plug-in Type to **IP Access Control**. A plug-in definition template in the YAML format is automatically loaded in the **Script Configuration** field. Modify template content.



Parameter	Description

Parameter	Description
type	◦ ALLOW: You can configure a whitelist to allow the API requests that meet specific requirements. The following types of whitelists are supported: ▪ You can configure a whitelist that contains only IP addresses. In this case, only API requests from the IP addresses in the whitelist are allowed. ▪ You can configure a whitelist that contains apps and their IP addresses. In this case, each app can send API requests only from its IP addresses in the whitelist. ◦ REFUSE: You can configure an IP address blacklist. API Gateway rejects all API requests from the IP addresses in the blacklist.

Script template of the IP address-based access control plug-in

```
---
type: ALLOW     # The type of access control. You can set this parameter to ALLOW to apply a whitelist o
r to REFUSE to apply a blacklist.
items:
- blocks:       # The IP address segment.
 - 78.11.12.2   # Specifies an IP address.
 - 61.3.9.0/24  # Specifies a CIDR block.
 appId: 219810  # Optional. If you specify this parameter, this IP address-based access control policy ap
plies only to the app specified by this parameter.
- blocks:       # The IP address segment.
 - 79.11.12.2   # Specifies an IP address.
```

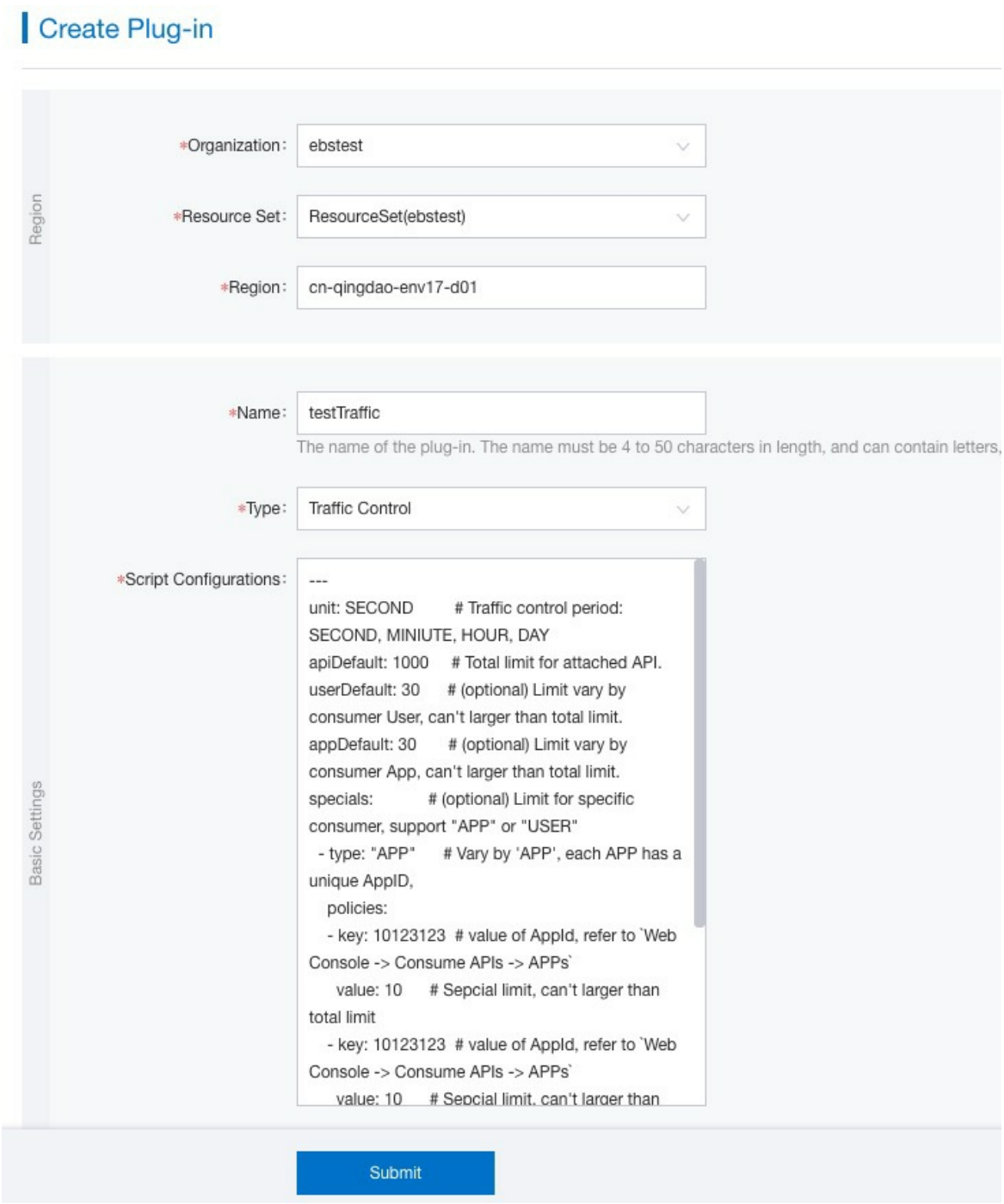
4. Click **Submit**.

1.5.4.2.2. Create a throttling plug-in

You can use a throttling plug-in to limit the number of API requests. A throttling plug-in helps prevent a backend service from being overwhelmed by a large number of API requests. This topic describes how to create a throttling plug-in.

Procedure

1. Log on to the API Gateway console .
2. In the left-side navigation pane, choose **Publish APIs > Plugin**.
3. On the Plugins list page, click **Create Plugin** in the upper-right corner. On the Create Plug-in page, specify **Organization**, **Resource Set**, and **Region**.Then, specify **Plug-in Name**, set Plug-in Type to **Traffic Control**, and modify configurations in the Script Configuration field.



Parameter	Description
unit	The unit of time. Valid values: SECOND, MINUTE, HOUR, and DAY.
apiDefault	The default API-level throttling threshold. It indicates the maximum number of times that an API bound with a throttling policy can be called within a specific unit of time. This parameter is set based on the backend service capability. This parameter is required.

Parameter	Description
userDefault	The default user-level throttling threshold. It indicates the maximum number of times that each user can call an API that is bound with a throttling policy within a specific unit of time. The user-level throttling threshold cannot be greater than the API-level throttling threshold. This parameter is optional.
appDefault	The default app-level throttling threshold. It indicates the maximum number of times that each app can call an API that is bound with a throttling policy within a specific unit of time. The app-level throttling threshold cannot be greater than the user-level throttling threshold. This parameter is optional.
specials	The special throttling settings. This parameter is optional. You can set throttling thresholds for special apps or users in a throttling policy. After this parameter is specified, the special throttling settings prevail for special apps or users.

Script template

```
---
unit: SECOND       # The unit of time. Valid values: SECOND, MINUTE, HOUR, and DAY.
apiDefault: 1000    # The default API-level throttling threshold.
userDefault: 30     # Optional. The default user-level throttling threshold. If you set this threshold to 0, user-level throttling is not performed. The user-level throttling threshold cannot be greater than the API-level throttling threshold.
appDefault: 30      # Optional. The default app-level throttling threshold. If you set this threshold to 0, app-level throttling is not performed. The app-level throttling threshold cannot be greater than the user-level throttling threshold.
specials:         # Optional. The special throttling settings. You can set throttling thresholds for special apps or users in a throttling policy.
 - type: "APP"     # The special throttling type. The value APP indicates that throttling is performed for special apps based on their AppKeys.
   policies:
   - key: 10123123  # The app ID. You can obtain the ID of an app from the app details page. To go to this page, choose Consume APIs > APPs in the left-side navigation pane of the API Gateway console and click the name of the app.
     value: 10     # The special throttling threshold for the app. This threshold cannot be greater than the user-level throttling threshold in the throttling policy.
   - key: 10123121  # The app ID.
     value: 10     # The special throttling threshold for the app. This threshold cannot be greater than the user-level throttling threshold in the throttling policy.
 - type: "USER"    # The special throttling type. The value USER indicates that throttling is performed for special Apsara Stack tenant accounts.
   policies:
   - key: 123455    # The ID of an Apsara Stack tenant account. You can move the pointer over the profile picture in the upper-right corner of the Alibaba Cloud Management Console to obtain the ID.
     value: 100     # The special throttling threshold for the Apsara Stack tenant account. This threshold cannot be greater than the API-level throttling threshold in the throttling policy.
```

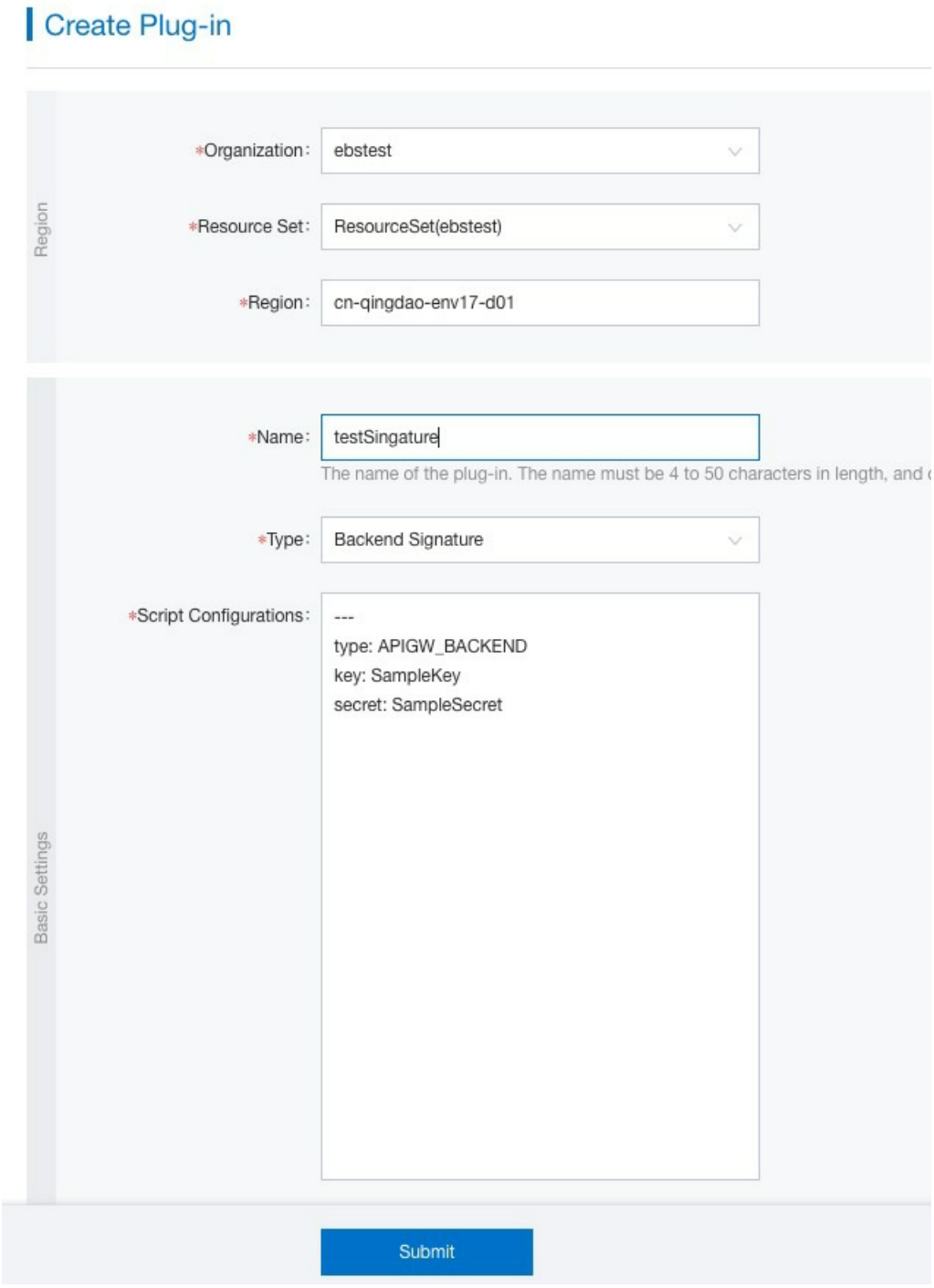
4. Click **Submit**.

1.5.4.2.3. Create a backend signature plugin

A backend signature plugin is used for signature verification between API Gateway and your backend service. A backend signature is a key-secret pair that you create and issue to API Gateway. It works in a way similar to an account and password pair. When API Gateway sends a request to your backend service, API Gateway uses the backend signature to calculate a signature string and pass it to your backend service. Your backend service obtains the signature string and authenticates API Gateway by using symmetric calculation. Perform the following steps to create a backend signature plugin:

Procedure

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > Plugin**.
3. On the Plugins list page, click **Create Plugin** in the upper-right corner. On the Create plugin page, specify **Organization**, **Resource Set**, **Region**, and **Name**. Set Type to **Backend Signature**.



Configure the plugin parameters as required.

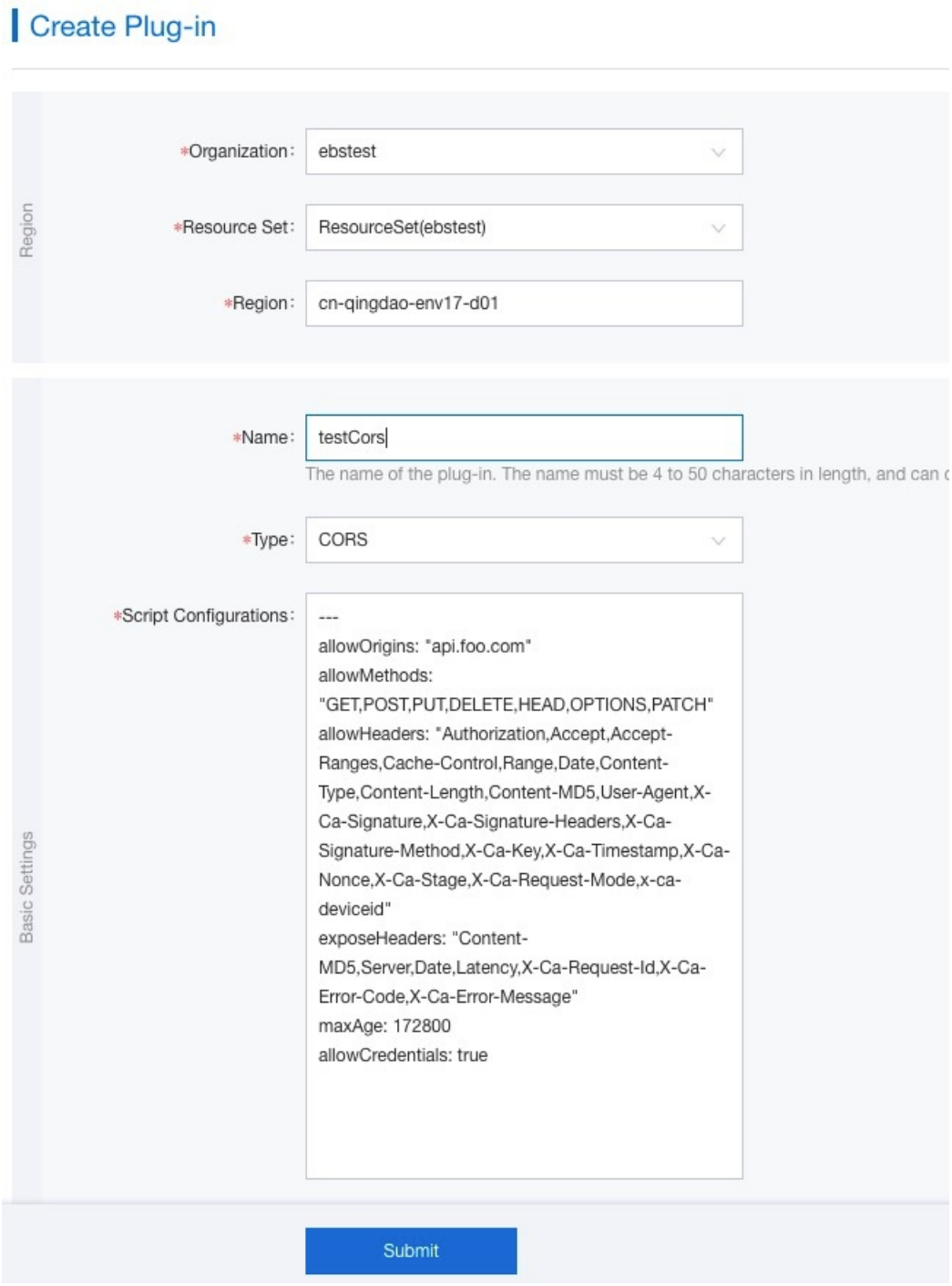
4. Click **Submit**.

1.5.4.2.4. Create a CORS plugin

This topic describes how to create a cross-origin resource sharing (CORS) plugin. If a resource requests another resource from a different domain or port of a different server, the former resource initiates a cross-domain HTTP request. For security purposes, the browser blocks the request and reports an error message. In this case, you need to use a CORS plugin to troubleshoot the issue.

Procedure

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > Plugin**.
3. On the Plugins list page, click **Create Plugin** in the upper-right corner. On the Create plugin page, specify **Organization**, **Resource Set**, **Region**, and **Name**. Set Type to **CORS**.



Cross-domain access template

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

Configure the plugin parameters as required.

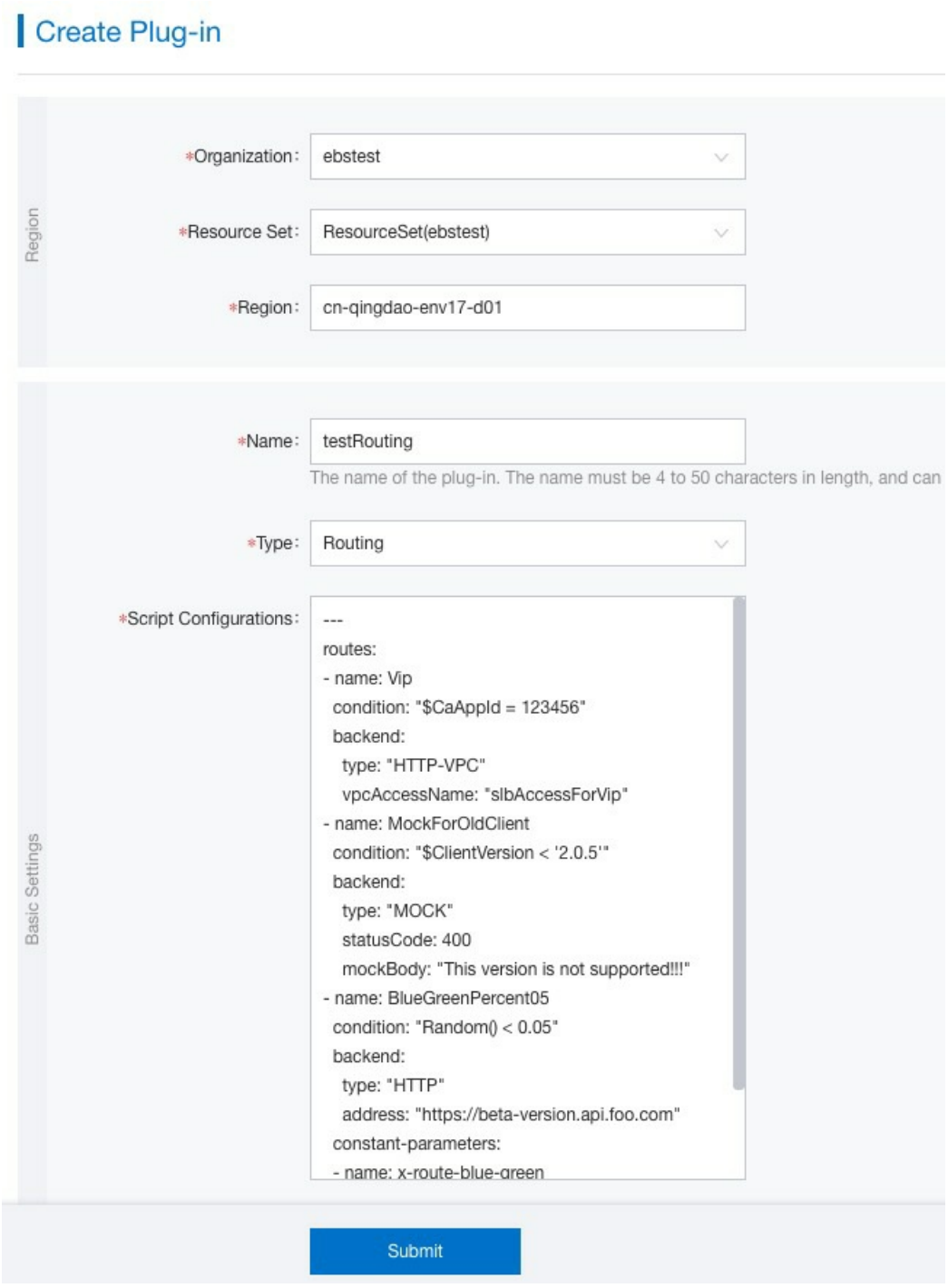
4. Click **Submit**.

1.5.4.2.5. Create a backend routing plug-in

A backend routing plug-in is used to route API requests to different backend services by changing the backend service type, backend service address, backend request path, and response parameters based on request and system parameters in API requests. Backend routing plug-ins can be used for multi-tenant routing and blue-green release. They can also be used to distinguish between different environments.

Create a backend routing plug-in

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > Plugin**.
3. On the Plugins list page, click **Create Plugin** in the upper-right corner. On the Create Plug-in page, specify **Organization**, **Resource Set**, **Region**, and **Plug-in Name**. Set Type to **Routing**.



4. Modify configurations in the Script Configuration field and click **Submit**.

Configurations

Configuration template

1. You can configure a backend routing plug-in in the JSON or YAML format because these two formats have the same schema. You can use the yaml to json tool to convert the plug-in configuration format. The following example describes a plug-in configuration template in the YAML format:

```
---
routes:
# Responses that are no longer supported are returned to clients of an earlier version. ClientVersion is a custom parameter in the API.
- name: MockForOldClient
 condition: "$ClientVersion < '2.0.5'"
 backend:
  type: "MOCK"
  statusCode: 400
  mockBody: "This version is not supported!!!"
# Blue-green release scenarios: Five percent of requests are routed to the backend of a blue-green release.
- name: BlueGreenPercent05
 condition: "Random() < 0.05"
 backend:   type: "HTTP"
  address: "https://beta-version.api.foo.com"
 constant-parameters:
 - name: x-route-blue-green
  location: header
  value: "route-blue-green"
```

The template has a root object routes that contains multiple route objects. Each route object is used to specify a routing rule. Each routing rule consists of the following parts:

- name: the name of the routing rule. The name must be unique within each plug-in and can contain only letters and digits. If an API request hits the rule, an HTTP header X-Ca-Routing-Name that contains the name of the rule is added to the request before the request is routed to your backend service.
- condition: the conditional expression of the routing rule. If an API request meets the condition, the request hits the routing rule. The backend routing plug-in checks the routing rules based on the order in which they are configured. The API request is routed to your backend service in the first routing rule that the request hits. After this occurs, the plug-in does not check the remaining routing rules. If you configure multiple routing rules, make sure that they are configured in the order that meets your service expectations.
- backend: the description of your backend service. The description must be consistent with the Swagger specification files for API Gateway. The backend configurations for an API in a backend routing plug-in override the original backend configurations in the API. If the backend configurations are incomplete after they are overridden, the X-Ca-Error-Code: I504RB error is reported to the client. If this error is returned, check whether your backend configurations are complete.
- constant-parameters: the constant parameters that you can customize in the routing rule. Constant parameters are included in an API request before the request is routed to your backend service. These parameters are used in the business logic of your backend service. A constant parameter can be a query or header parameter.

Conditional expressions

Basic syntax

- The syntax of conditional expressions in backend routing plug-ins are similar to that of SQL statements. The basic format is $A = 'A' and '$B = 'B'.
- Each parameter starts with $. You can reference the request parameters that are defined in an API to

which a plug-in is bound. The request mode of the API can be set to Request Parameter Mapping(Filter Unknown Parameters), Request Parameter Mapping(Passthrough Unknown Parameters), or Request Parameter Passthrough. If you define a request parameter named query1 when you configure an API, you can use $query1 to reference this parameter in conditional expressions.

- The following constant parameter types are supported:
  - STRING: the string data type. Single or double quotation marks can be used to enclose a string. Example: "Hello".
  - INTEGER: the integer data type. Example: 1001 and -1.
  - NUMBER: the floating point data type. Example: 0.1 and 100.0.
  - BOOLEAN: the Boolean data type. Valid values: true and false.
- You can use and and or operators to connect different expressions.
- You can use parentheses () to specify the priority of conditional expressions.
- Random() is a built-in function. It generates a NUMBER-type parameter that returns a random number in the range of [0, 1).
- You can use $CaAppId to reference system parameters of the current request. You can reference system parameters without the need to define them in an API. However, if you have defined a parameter in the API with the same name as a system parameter, the value of the system parameter is overwritten by that of the defined parameter. The following system parameters apply to backend routing plug-ins:
  - CaStage: the environment to which the requested API is published. Valid values: RELEASE, PRE, and TEST.
  - CaDomain: the domain name of the API group to which the requested API belongs.
  - CaRequestHandleTime: the time in UTC at which the current request is received.
  - CaAppId: the value of the AppId parameter in the current request.
  - CaAppKey: the value of the AppKey parameter in the current request.
  - CaClientIp: the IP address of the client from which the current request is sent.
  - CaApiName: the name of the requested API.
  - CaHttpScheme: the protocol used by the current request. Valid values: HTTP, HTTPS, and WS.
  - CaClientUa: the UserAgent field uploaded from the client.
- If you use an unknown parameter in a conditional expression, such as $UnknonwParameter = 1, the result of the expression is false.

Conditional expression examples

- The following expression indicates that the probability is less than 5%:

```
Random() < 0.05
```

- The following expression indicates that the requested API is published to the Test environment:

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

- The following expression indicates that the custom parameter UserName is set to Admin and the source IP address is 47.47.74.77.

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

- The following expression indicates that the AppId parameter is set to 1001, 1098, or 2011, and the protocol that is used by the API request is HTTPS:

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

Backend configuration and overriding rules

The structure of a backend service is consistent with the Swagger definitions imported into API Gateway. The following examples show the supported backend service types and configuration samples. The backend configurations in a backend routing plug-in override the backend configurations in an API that is bound to the plug-in. If you do not need to change the backend service type, specify only the parameters whose values you want to change.

- HTTP

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

- HTTP-VPC

```
---
backend:
 type: HTTP-VPC
 vpcId: vpc-xxxx
 vpcInstance: 172.168.1.1
 vpcInstancePort: 80
 path: "/users/{userId}"
 method: GET
 timeout: 10000
```

- MOCK

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

Limits

- The metadata of a backend routing plug-in can be a maximum of 16,384 bytes in size. If this limit is exceeded, the InvalidPluginData.TooLarge error is reported.
- A maximum of 16 routing rules can be configured in a backend routing plug-in. If this limit is exceeded, the InvalidPluginData.TooManyRoutes error is reported.

- The size of a single conditional expression cannot exceed 512 bytes. If this limit is exceeded, the InvalidPluginData.ConditionTooLong error is reported.
- Configuration updates in a plug-in are synchronized in real time to all the APIs bound to the plug-in. An interval of at least 45s is required between two updates. If you update a plug-in twice within less than 45s, the InvalidPluginData.UpdateTooBusy error is reported.

Typical scenarios

- Configure multi-tenant routing. Different backend service addresses are allocated based on the AppId settings. For example, users whose app ID is 10098 or 10099 are VIP customers. API requests from these users are required to be routed to an independent server cluster.

```
---
-routes:
# If the AppId value for an API caller is 10098 or 10099, requests to the API are routed to an independent ad
dress.
# In this example, the VPC access name is set to slbAddressForVip.
- name: http1
 condition: "$CaAppId = 10098 or $CaAppId = 10099"
 backend:
  type: "HTTP"
  address: "https://test-env.foo.com"
```

- Configure routing based on environment settings (Test, Pre, and Release). All requests for the APIs that are published to the same environment are required to be routed to the same server.

```
---
routes:
# Route all requests for APIs that are published to the Test environment to the test server on the Internet.
- name: Vip
 condition: "$CaStage = 'TEST'"
 backend:
  type: "HTTP"
  address: "https://test-env.foo.com"
```

- Perform a blue-green release. Five percent of requests are required to be directed to a group of beta servers to perform a blue-green release.

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

1.5.4.2.6. Create a caching plugin

You can bind a caching plugin to an API to cache the responses from your backend service. This reduces the load on the backend service and shortens the response time.

1. Usage notes

- Caching plugins can cache only the responses to API requests that use the GET method.
- When you configure a caching plugin, you can use the following parameters to sort responses in a cache:
  - varyByApp: controls whether to match and serve cached responses based on the app IDs of API callers.
  - varyByParameters: controls whether to match and serve cached responses based on the values of specific parameters. The plugin uses the same request parameters of APIs that are bound to the plugin to sort the responses to API requests.
  - varyByHeaders: controls whether to match and serve cached responses based on different request headers. For example, match and serve cached responses based on the `Accept` or `Accept-Language` header.
- API Gateway provides each user with 5 MB of cache space in each region. Caches are cleared after expiration. If a cache reaches its space limit, no more responses are stored in the cache.
- If `Cache-Control` is specified in a response from your backend service, the response is stored in a cache based on the specified cache policy. If Cache-Control is not specified in a response, after the response expires, the response is stored in a cache based on the default cache policy and is stored for the period of time specified by the duration parameter.
- A response can be stored in a cache for a maximum of 48 hours (172,800 seconds) after it expires. Configurations made after the 48 hours are invalid.
- API Gateway determines how to process the `Cache-Control` headers of client requests based on the clientCacheControl settings. By default, API Gateway does not process the Cache-Control headers. You can set clientCacheControl to the following modes:
  - off: API Gateway ignores the `Cache-Control` headers of all client requests.
  - all: API Gateway processes the `Cache-Control` headers of all client requests.
  - app: API Gateway processes only the `Cache-Control` headers of client requests whose `app IDs` are included in the configured `apps` list.
- By default, API Gateway caches only the `Content-Type`, `Content-Encoding`, and `Content-Language` headers in responses. If you need to cache more headers, add the headers in the `cacheableHeaders` parameter of the caching plugin.

2. Configurations

You can configure a caching plugin in the JSON or YAML format because these two formats have the same schema. You can use the yaml to json tool to convert the plugin configuration format. The following example describes a plugin configuration template in the YAML format:

```
---
varyByApp: false   # Controls whether to match and serve cached responses based on the app IDs of API callers. Default value: false.
varyByParameters:  # Controls whether to match and serve cached responses based on the values of specific parameters.
- userId        # The name of a backend parameter. If the backend parameter is mapped to a parameter with a different name, set this parameter to the mapped parameter name.
varyByHeaders:    # Controls whether to match and serve cached responses based on different request headers.
- Accept        # Cached responses are matched and served based on the Accept header.
clientCacheControl: # API Gateway determines how to process the Cache-Control headers of client requests based on the clientCacheControl settings.
 mode: "app"     # Valid values: off, all, and apps. Default value: off. off indicates that API Gateway ignores the Cache-Control headers of all client requests. all indicates that API Gateway processes the Cache-Control headers of all client requests. apps indicates that API Gateway processes only the Cache-Control headers of client requests whose app IDs are included in the configured apps list.
 apps:         # A list of app IDs. If mode is set to app, API Gateway processes only the Cache-Control headers of client requests whose app IDs are in this list.
 - 1992323      # A sample app ID. It is not an AppKey.
 - 1239922      # A sample app ID. It is not an AppKey.
cacheableHeaders:  # The cacheable response headers. By default, API Gateway caches only the Content-Type and Content-Length headers of backend responses.
- X-Customer-Token  # The name of the cacheable response header.
duration: 3600    # The default grace period, in seconds.
```

3. Working mechanism

If an API request hits the cache of an API, the `X-Ca-Caching: true` header is included in the response to the API request.

4. Limits

- The metadata of a caching plugin can be a maximum of 16,380 bytes in size.
- A response body that exceeds 128 KB in size cannot be cached.
- Each user has a maximum of 30 MB of total cache space in each region.

1.5.4.2.7. JWT authentication plug-in

RFC 7519-compliant JSON Web Token (JWT) is a simple method used by API Gateway to authenticate requests. `API Gateway` hosts the `public JSON Web Keys (JWKs)` of users and uses these JWKs to sign and authenticate JWTs in requests. Then, API Gateway forwards `claims` to backend services as backend parameters. This simplifies the development of backend applications.

Compared with the `OpenID Connect` feature, the `JWT authentication plug-in` can implement the functions of this feature and bring the following benefits:

- You do not need to configure an additional `authorization API`. `JWTs` can be generated and distributed in multiple ways. API Gateway is only responsible for `JWT` authentication by using `public JWKs`.
- `JWKs` without `kid` specified are supported.
- Multiple `JWKs` can be configured.

- You can read token information from the `header` of a request or a `query` parameter.
- If you want to transmit a `JWT` in an Authorization header, such as `Authorization bearer {token}`, you can set `parameter` to Authorization and `parameterLocation` to header so that the token information is correctly read.
- The `jti` claim-based anti-replay check is supported if you set `preventJtiReplay` to true.
- Requests that do not include tokens can be forwarded to backend services without verification if you set `bypassEmptyToken` to true.
- The verification on the `exp` setting for tokens can be skipped if you set `ignoreExpirationCheck` to true.

If you configure a `JWT authentication plug-in` and bind it to an `API` for which the `OpenID Connect` feature is configured, the JWT authentication plug-in takes effect in place of the `OpenID Connect` feature.

1. Obtain a JWK

RFC 7517-compliant JWK is used to sign and authenticate JWTs. If you want to configure a `JWT authentication plug-in`, you must generate a valid `JWK` manually or by using an online `JWK generator` such as mkjwk.org. The following example shows a valid `JWK`. In the JWK example, the private key is used to sign the token, and the public key is configured in the `JWT authentication plug-in` to authenticate the signature.

```
{
 "kty": "RSA",
 "e": "AQAB",
 "kid": "O9fpdhrViq2zaaaBEWZITz",
 "use": "sig",
 "alg": "RS256",
 "n": "qSVxcknOm0uCq5vGsOmaorPDzHUubBmZZ4UXj-9do7w9X1uKFXAnqfto4TepSNuYU2bA_-tzSLAGBsR-BqvT6w9SjxakeiyQpVmexxnDw5WZwpWenUAcYrfSPEoNU-0hAQwFYgqZwJQMN8ptxkd0170PFauwACOx4Hfr-9FPGy8NCoIO4MfLXzJ3mJ7xqgIZp3NIOGXz-GIAbCf13ii7kSStpYqN3L_zzpvXUAos1FJ9IPXRV84tIZpFVh2lmRh0h8ImK-vI42dwlD_hOIzayL1Xno2R0T-d5AwTSdnep7g-Fwu8-sj4cCRWq3bd61Zs2QOJ8iustH0vSRMYdP5oYQ"
}
```

*The preceding JWK is in the JSON format. If you want to configure a JWT authentication plug-in in the YAML format, you must use a JWK in the YAML format.**

- For a `JWT authentication plug-in`, you need only to configure a `public key`. Keep your `private key` safe. The following table lists the signature algorithms supported by the JWT authentication plug-in.

Signature algorithm	Supported `alg` setting
RSASSA-PKCS1-V1_5 with SHA-2	RS256, RS384, RS512
Elliptic Curve (ECDSA) with SHA-2	ES256, ES384, ES512
HMAC using SHA-2	HS256, HS384, HS512

When you configure a key of the HS256, HS384, or HS512 type, the key value is base64url encoded. If the signature is invalid, check whether your key is in the same format as the key used to generate the token.

2. Plug-in configurations

You can configure a JWT authentication plug-in in the JSON or YAML format because these two formats have the same schema. You can use the `yaml to json` tool to convert the plug-in configuration format. The following example shows a plug-in configuration template in the YAML format:

```
---
parameter: X-Token        # The parameter from which the JWT is read. It corresponds to a parameter in an API request.
parameterLocation: header   # The location from which the JWT is read. Valid values: query and header. This parameter is optional if Request Mode for the bound API is set to Request Parameter Mapping(Filter Unknown Parameters) or Request Parameter Mapping(Passthrough Unknown Parameters). This parameter is required if Request Mode for the bound API is set to Request Parameter Passthrough.
preventJtiReplay: false    # Controls whether to enable the anti-replay check for jti. Default value: false.
bypassEmptyToken: false    # Controls whether to forward requests that do not include tokens to backend services without verification.
ignoreExpirationCheck: false # Controls whether to ignore the verification of the exp setting.
claimParameters:          # The claims to be converted into parameters. API Gateway maps JWT claims to backend parameters.
- claimName: aud          # The name of the JWT claim, which can be public or private.
  parameterName: X-Aud     # The name of the backend parameter, to which the JWT claim is mapped.
  location: header        # The location of the backend parameter, to which the JWT claim is mapped. Valid values: query, header, path, and formData.
- claimName: userId        # The name of the JWT claim, which can be public or private.
  parameterName: userId     # The name of the backend parameter, to which the JWT claim is mapped.
  location: query         # The location of the backend parameter, to which the JWT claim is mapped. Valid values: query, header, path, and formData.
#
# Public key in the JWK
jwk:
  kty: RSA
  e: AQAB
  use: sig
  alg: RS256
  n: qSVxcknOm0uCq5vGsOmaorPDzHUubBmZZ4UXj-9do7w9X1uKFXAnqfto4TepSNuYU2bA_-tzSLAGBsR-BqvT6w9SjxakeiyQpVmexxnDw5WZwpWenUAcYrfSPEoNU-0hAQwFYgqZwJQMN8ptxkd0170PFauwACOx4Hfr-9FPGy8NCoIO4MfLXzJ3mJ7xqgIZp3NIOGXz-GIAbCf13ii7kSStpYqN3L_zzpvXUAos1FJ9IPXRV84tIZpFVh2lmRh0h8ImK-vI42dwlD_hOIzayL1Xno2R0T-d5AwTSdnep7g-Fwu8-sj4cCRWq3bd61Zs2QOJ8iustH0vSRMYdP5oYQ
#
# You can configure multiple JWKs and use them together with the jwk field.
# If multiple JWKs are configured, kid is required. If the JWT does not include kid, the consistency check on kid fails.
jwks:
- kid: O9fpdhrViq2zaaaBEWZITz  # If only one JWK is configured, kid is optional. If the JWT includes kid, API Gateway checks the consistency of kid.
  kty: RSA
  e: AQAB
  use: sig
  alg: RS256
  n: qSVxcknOm0uCq5v....
- kid: 10fpdhrViq2zaaaBEWZITz  # If only one JWK is configured, kid is optional. If the JWT includes kid, API Gateway checks the consistency of kid.
  kty: RSA
  e: AQAB
  use: sig
  alg: RS256
  n: qSVxcknOm0uCq5v...
```

- The `JWT authentication plug-in` retrieves JWTs based on the settings of `parameter` and `parameterLocation`. For example, if `parameter` is set to X-Token and `parameterLocation` is set to header, the JWT is read from the `X-Token` header.
- If the parameter configured in an API has the same name as the parameter specified by `parameter`, do not specify `parameterLocation`. Otherwise, an error is reported when the API is called.
- If you want to transmit a token in an Authorization header, such as `Authorization bearer {token}`, you can set `parameter` to Authorization and `parameterLocation` to header so that the token information can be correctly read.
- If `preventJtiReplay` is set to true, the JWT authentication plug-in uses `jti` in `claims` to perform an anti-replay check.
- If `bypassEmptyToken` is set to true and a token is not included in a request, API Gateway skips the check and directly forwards the request to a backend service.
- If `ignoreExpirationCheck` is set to true, API Gateway skips the verification of the `exp` setting. Otherwise, API Gateway checks whether a token expires.
- If API Gateway is required to forward `claims` in tokens to backend services, you can set `tokenParameters` to configure the following parameters to be forwarded:
  - `claimName`: the name of the claim in a token, which can be `kid`.
  - `parameterName`: the name of the parameter forwarded to a backend service.
  - `location`: the location of the parameter forwarded to a backend service. Valid values: `header`, `query`, `path`, and `formData`.
    - If this parameter is set to `path`, the backend path must contain a parameter with the same name, such as `/path/{userId}`.
    - If this parameter is set to `formData`, the body of a received request in a backend service must be of the `Form` type.
- You can configure only one key in the `jwk` field. You can also configure multiple keys in the `jwks` field.
  - You can configure only one key with `kid` not specified.
  - You can configure multiple keys with `kid` specified. `kid` must be unique.

3. Verification rules

- A JWT authentication plug-in obtains tokens based on the settings of `parameter` and `parameterToken`. If API Gateway is required to forward requests to backend services even when tokens are not included in the requests, set `bypassEmptyToken` to true.
- If you want to configure multiple keys, abide by the following principles:
  - Preferentially select a key whose ID is the same as the value of `kid` in a token for signature and authentication.
  - You can configure only one key with `kid` not specified. If no key whose ID is the same as the value of `kid` in a token exists, use the key with `kid` not specified for signature and authentication.
  - If all the configured keys have specified `kid` settings, and the token in a request does not contain `kid` or no keys match `kid`, an `A403JK` error is reported.
- If a token contains `iat`, `nbf`, and `exp`, the JWT authentication plug-in verifies the validity of their time formats.
- By default, API Gateway verifies the setting of `exp`. If you want to skip the verification, set `ignoreExpirationCheck` to true.

- `tokenParameters` is configured to extract the required parameters from the `claims` of a token. These parameters are forwarded to backend services.

4. Configuration examples

4.1 Configure a single JWK

```
---
parameter: X-Token       # The parameter from which the JWT is read. It corresponds to a parameter in an API request.
parameterLocation: header  # The location from which the JWT is read. Valid values: query and header. This parameter is optional if Request Mode for the bound API is set to Request Parameter Mapping(Filter Unknown Parameters) or Request Parameter Mapping(Passthrough Unknown Parameters). This parameter is required if Request Mode for the bound API is set to Request Parameter Passthrough.
claimParameters:         # The claims to be converted into parameters. API Gateway maps JWT claims to backend parameters.
- claimName: aud         # The name of the JWT claim, which can be public or private.
  parameterName: X-Aud    # The name of the backend parameter, to which the JWT claim is mapped.
  location: header       # The location of the backend parameter, to which the JWT claim is mapped. Valid values: query, header, path, and formData.
- claimName: userId      # The name of the JWT claim, which can be public or private.
  parameterName: userId   # The name of the backend parameter, to which the JWT claim is mapped.
  location: query        # The location of the backend parameter, to which the JWT claim is mapped. Valid values: query, header, path, and formData.
preventJtiReplay: false   # Controls whether to enable the anti-replay check for jti. Default value: false.
#
# Public key in the JWK
jwk:
  kty: RSA
  e: AQAB
  use: sig
  alg: RS256
  n: qSVxcknOm0uCq5vGsOmaorPDzHUubBmZZ4UXj-9do7w9X1uKFXAnqfto4TepSNuYU2bA_-tzSLAGBsR-BqvT6w9SjxakeiyQpVmexxnDw5WZwpWenUAcYrfSPEoNU-0hAQwFYgqZwJQMN8ptxkd0170PFauwACOx4Hfr-9FPGy8NCoIO4MfLXzJ3mJ7xqgIZp3NIOGXz-GIAbCf13ii7kSStpYqN3L_zzpvXUAos1FJ9IPXRV84tIZpFVh2lmRh0h8ImK-vI42dwlD_hOIzayL1Xno2R0T-d5AwTSdnep7g-Fwu8-sj4cCRWq3bd61Zs2QOJ8iustH0vSRMYdP5oYQ
```

4.2 Configure multiple JWKs

```
---
parameter: Authorization  # The parameter from which the token is obtained.
parameterLocation: header  # The location from which the token is obtained.
claimParameters:        # The claims to be converted into parameters. API Gateway maps JWT claims to backend parameters.
- claimName: aud        # The name of the JWT claim, which can be public or private.
  parameterName: X-Aud    # The name of the backend parameter, to which the JWT claim is mapped.
  location: header       # The location of the backend parameter, to which the JWT claim is mapped. Valid values: query, header, path, and formData.
- claimName: userId      # The name of the JWT claim, which can be public or private.
  parameterName: userId   # The name of the backend parameter, to which the JWT claim is mapped.
  location: query        # The location of the backend parameter, to which the JWT claim is mapped. Valid values: query, header, path, and formData.
preventJtiReplay: true    # Controls whether to enable the anti-replay check for jti. Default value: false.
jwks:
- kid: O9fpdhrViq2zaaaBEWZITz  # kid must be set to different values for different JWKs.
  kty: RSA
  e: AQAB
  use: sig
  alg: RS256
  n: qSVxcknOm0uCq5v....
- kid: 10fpdhrViq2zaaaBEWZITz  # kid must be set to different values for different JWKs.
  kty: RSA
  e: AQAB
  use: sig
  alg: RS256
  n: qSVxcknOm0uCq5v...
```

5. Error codes

HTTP status code	Error code	Error message	Description
400	I400JR	JWT required	The error message returned because no JWT-related parameters are found.
403	S403JI	Claim `jti` is required when `preventJtiReplay:true`	The error message returned because no valid `jti` claims are included in the request when preventJtiReplay is set to true in a `JWT authentication plug-in`.

HTTP status code	Error code	Error message	Description
403	S403JU	Claim `jti` in JWT is used	The error message returned because the `jti` claim that is included in the request has been used when preventJtiReplay is set to true in a **JWT authentication plug-in**.
403	A403JT	Invalid JWT: ${Reason}	The error message returned because the **JWT** that is read from the request is invalid.
400	I400JD	JWT Deserialize Failed: **${Token}**	The error message returned because the **JWT** that is read from the request fails to be parsed.
403	A403JK	No matching JWK, **kid:${kid}** not found	The error message returned because no **JWK** matches **kid** configured in the **JWT** included in the request.
403	A403JE	JWT is expired at **${Date}**	The error message returned because the **JWT** that is read from the request expires.
400	I400JP	Invalid JWT plugin config: ${JWT}	The error message returned because the **JWT authentication plug-in** is incorrectly configured.

If an HTTP response message includes an unexpected response code specified by `ErrorCode` in the `X-Ca-Error-Code` header, such as `A403JT` or `I400JD`, you can visit the jwt.io website to check the token validity and format.

6. Limits

- The metadata of a JWT authentication plug-in can contain a maximum of **16,380** characters.
- You can configure a maximum of **16** parameters to be forwarded. Both the **claimName** and **parameterName** parameters cannot exceed 32 characters in length. Only the following regular expression is supported: [A-Za-z0-9-_].

- `alg` can be set to RS256, RS384, RS512, ES256, ES384, ES512, HS256, HS384, or HS512 for JWKs.

1.5.4.2.8. Access control plugin

1. Overview

In an access control plugin, you can define conditions based on the request parameters or context of an API to which the plugin is bound. This allows you to determine whether to deliver an API request to a backend service. For information about how to define parameters and use conditional expressions, see Use parameters and conditional expressions.

2. Configurations

Assume that the API request path is `/{userId}/...`. JWT authentication is enabled for APIs. Two claims, userId and userType, are available in the JWT. The following plugin verification conditions apply:

- If userType is set to admin, requests in all paths are allowed.
- If userType is set to user, only the requests in the same /{userId} path are allowed.

```
---
#
# Assume that the API request path is /{userId}/... in this example.
# JWT authentication is enabled for APIs. Two claims, userId and userType, are available in the JWT.
# The following plugin verification conditions apply:
# - If userType is set to admin, requests in all paths are allowed.
# - If userType is set to user, only the requests in the same /{userId} path are allowed.
parameters:
 userId: "Token:userId"
 userType: "Token:userType"
 pathUserId: "path:userId"
#
# Rules are defined based on the preceding parameters. For each API request, the plugin checks the rules in s
equence. If a condition in a rule is met, the result is true and the action that is specified by ifTrue is performe
d. If a condition in a rule is not met, the result is false and the action that is specified by ifFalse is performed.
# The action ALLOW indicates that the request is allowed. The action DENY indicates that the request is deni
ed and an error code is returned to the client. After the ALLOW or DENY action is performed, the plugin does
not check the remaining conditions.
# If neither the ALLOW nor DENY action is performed, the plugin proceeds to check the next condition.
rules:
 - name: admin
   condition: "$userType = 'admin'"
   ifTrue: "ALLOW"
 - name: user
   condition: "$userId = $pathUserId"
   ifFalse: "DENY"
   statusCode: 403
   errorMessage: "Path not match ${userId} vs /${pathUserId}"
   responseHeaders:
    Content-Type: application/xml
   responseBody:
    <Reason>Path not match ${userId} vs /${pathUserId}</Reason>
```

3. Relevant errors

Error code	HTTP status code	Message	Description
A403AC	403	Access Control Forbidden by ${RuleName}	The error message returned because the request is rejected by the access control plugin that is bound to the API.

4. Limits

- A maximum of **16** parameters can be specified in an access control plugin.
- Each conditional expression can contain a maximum of **512** characters.
- The metadata of an access control plugin can contain a maximum of **16,380** characters.
- A maximum of 16 **rules** can be configured in each access control plugin.

1.5.4.2.9. Error code mapping plug-in

An **error code mapping plug-in** is used to map backend error responses to expected error responses based on mapping rules that are defined by clients.

1. Overview

An error code mapping plug-in is used to map backend error responses to expected error responses based on mapping rules that are defined by clients.

2. Quick start

The following example shows an error response that is returned by a backend service. The HTTP status code is 200, but the response body contains an error message in a JSON string.

```
HTTP 200 OK
Content-Type:application/json
{"req_msg_id":"d02afa56394f4588832bed46614e1772","result_code":"ROLE_NOT_EXISTS"}
```

- Clients want to receive an HTTP status code other than 200 but do not want to modify backend configurations. The clients expect the following sample error response:

```
HTTP 404
X-Ca-Error-Message: Role Not Exists, ResultId=d02afa56394f4588832bed46614e1772
```

In this case, you can use the following sample to configure an error code mapping plug-in and bind the plug-in to the related APIs:

```
---
# The parameters that are involved in a mapping.
parameters:
 statusCode: "StatusCode"
 resultCode: "BodyJsonField:$.result_code"
 resultId: "BodyJsonField:$.req_msg_id"
# The mapping condition.
errorCondition: "$statusCode = 200 and $resultCode <> 'OK'"
# The parameter in an error response that is used to specify the error code and hit mapping rules.
errorCode: "resultCode"
# Mapping rules.
mappings:
 - code: "ROLE_NOT_EXISTS"
   statusCode: 404
   errorMessage: "Role Not Exists, RequestId=${resultId}"
 - code: "INVALID_PARAMETER"
   statusCode: 400
   errorMessage: "Invalid Parameter, RequestId=${resultId}"
# Optional. The default mapping rule.
defaultMapping:
 statusCode: 500
 errorMessage: "Unknown Error, ${resultCode}, RequestId=${resultId}"
```

In this example, the HTTP status code and the `result_code` parameter in an error response are used to define the mapping condition. If the HTTP status code of an error response is 200 but the value of the `result_code` parameter is not `'OK'`, the mapping starts. The `result_code` parameter is used to define the mapping rules. If the value of the result_code parameter is `ROLE_NOT_EXISTS`, the original HTTP status code is mapped to 404. If the value of the result_code parameter is `INVALID_PARAMETER`, the original HTTP status code is mapped to 400. If the value of the result_code parameter is neither of the preceding values, the original HTTP status code is mapped to 500.

3. Plug-in configurations and mapping rules

3.1 Plug-in configurations

You can configure an error code mapping plug-in in the `JSON` or `YAML` format. The following parameters can be specified:

- `parameters`: required. The parameters that are involved in a mapping. These parameters are specified as key-value pairs in the `map` format. For information about how to define parameters and write conditional expressions, see Use parameters and conditional expressions.
- `errorCondition`: required. The condition under which a response is considered an error response. If the result of the conditional expression is `true`, the mapping starts.

- `errorCode`: optional. The parameter that is used to specify the error code in an error response and hit mapping rules. The error code that is specified by this parameter is compared with the value of the `code` parameter in the mapping rules specified by `mappings`.
- `mappings`: required. The mapping rules. API Gateway reconstructs error responses based on the setting of `errorCode` or `errorCondition`. A mapping rule may contain the following parameters:
  - `code`: optional. The value of this parameter must be unique among all mapping rules. If the error code of an error response is the same as the value of the `code` parameter in the current mapping rule, the error response is mapped based on the current mapping rule.
  - `condition`: optional. The condition under which an error response needs to be mapped based on the current mapping rule. If the result of the conditional expression is `true`, the error response is mapped based on the current mapping rule.
  - `statusCode`: required. The HTTP status code that replaces the original HTTP status code of an error response if the error response needs to be mapped based on the current mapping rule.
  - `errorMessage`: optional. The error message that is returned to the client after a mapping. The value of this parameter is obtained from the parameters in the original backend error response and is also stored in the `errorMessage` parameter in error logs. In the error response after the mapping, this parameter is displayed as the value of the `X-Ca-Error-Message` header.
  - `responseHeaders`: optional. The response headers that are included in an error response after a mapping if the current mapping rule is hit. This parameter is specified as key-value pairs in the map format.
  - `responseBody`: optional. The response body that overwrites the original response body of an error response if the error response needs to be mapped based on the current mapping rule.
- `defaultMapping`: optional. The default mapping rule. If all the rules that are defined in `mappings` are not hit by an error response, the error response is mapped based on this default mapping rule.
  - `statusCode`: required. The HTTP status code that replaces the original HTTP status code of an error response if the error response needs to be mapped based on the current mapping rule.
  - `errorMessage`: optional. The error message that is returned to the client after a mapping. The value of this parameter is obtained from the parameters in the original backend error response and is also stored in the `errorMessage` parameter in error logs. In the error response after the mapping, this parameter is displayed as the value of the `X-Ca-Error-Message` header.

- `responseHeaders`: optional. The response headers that are included in an error response after a mapping if the current mapping rule is hit. This parameter is specified as key-value pairs in the map format.
- `responseBody`: optional. The response body that overwrites the original response body of an error response if the error response needs to be mapped based on the current mapping rule.

Take note of the following points when you configure an error code mapping plug-in:

- The parameters that are used to write conditional expressions in `mappingCondition` and `mappings[].condition` must be defined in `parameters`. Otherwise, the plug-in does not work and reports an error. For information about how to define parameters and write conditional expressions, see Use parameters and conditional expressions.
- The value of the `errorCode` parameter must be the name of a parameter that is defined in `parameters`.
- When you configure a mapping rule specified by `mappings`, you must specify `code` or `condition`. When you specify `code`, the value of this parameter must be unique among all mapping rules. When you specify `condition`, you must write conditional expressions in the order that meets your requirements. This is because the order of conditions determines their priorities.
- You can specify `errorMessage` and `responseBody` in a format similar to `"${Code}: ${Message}"` and obtain the parameter values from those specified in `parameters`.
- You can specify `responseHeaders` in the `${Message}` format.
- If you do not specify `responseBody`, the body of an error response returned to the client after a mapping is the same as that of the original error response.
- You can use the `responseHeaders` parameter to specify headers and their settings to replace corresponding headers in a backend error response. If you specify the value of a header as `''`, this header will be deleted after a mapping. If you do not specify this parameter, the headers of the error response returned to the client after the mapping are the same as those of the original error response.
- If you do not specify `defaultMapping`, the error code mapping does not take effect. The original error response from your backend service is returned to the client.

3.2 Parameters involved in a mapping

As described in the following code, you must specify the parameters that are involved in a mapping as key-value pairs in `parameters`. Each key is the name of a parameter. Each value is specified in the `Location:Name` format. This format indicates that the value of a parameter is obtained from a specific location in the response or system context.

```
---
# The parameters that are involved in a mapping.
parameters:
 statusCode: "StatusCode"
 resultCode: "BodyJsonField:$.result_code"
 resultId: "BodyJsonField:$.req_msg_id"
```

An error code mapping plug-in supports the parameters at specific locations in the following table.

Location	Included in	Description
StatusCode	Response	The HTTP status code in a backend error response, such as `200` or `400`.
ErrorCode	Response	The error code of a system error response in API Gateway.
ErrorMessage	Response	The system error message in API Gateway.
Header	Response	Use `Header:{Name}` to obtain the first value of the HTTP header that is specified by `{Name}`.
BodyJsonField	Response*	Use `BodyJsonField:{JPath}` to obtain the JSON string in the body of an API request or a backend response in `JSONPath` mode.
System	Response	Use `System:{Name}` to obtain the value of the system parameter that is specified by `{Name}`.

Location	Included in	Description
Token	Response	If `JWT` is used with `OAuth2` for authentication, use `Token:{Name}` to obtain the value of the parameter that is specified by `{Name}` in a token.

- `ErrorCode` and `ErrorMessage` are used to return system error codes and detailed system error information in API Gateway. For more information, see Error codes.
- `BodyJsonField` can be used to obtain the JSON string in the body of a backend response. However, if the size of the response body exceeds **15,360** bytes, the string obtained is `null`.

3.3 Working mechanism

The following operations describe how an error code mapping plug-in works:

1. The plug-in obtains the values of the parameters from a backend error response and the system context based on the list of parameters that are defined in `parameters`.
2. The plug-in uses the parameters and obtained values to execute the conditional expression that is written in `errorCondition`. If the result is `true`, go to the next step. If the result is `false`, the process ends.
3. If `errorCode` is specified, the plug-in obtains the value of `errorCode`. Then, the plug-in checks whether a mapping rule exists, which indicates that the errorCode setting is the same as the setting of `code`. The mapping rule is specified by `mappings`.
4. If no mapping rule meets requirements, the plug-in executes in sequence the conditional expressions that are written in `condition` in mapping rules.
5. If a mapping rule is hit in Step 3 or Step 4, the original error response is mapped based on the mapping rule. Otherwise, the original error response is mapped based on the default mapping rule.

3.4 Mapping of system error codes and error logs

- In API Gateway, system errors may occur in processes such as check, verification, throttling, and plug-in operations. For more information, see Error codes. You can use `ErrorCode` as a location to obtain information in a system error response. For example, clients support only HTTP status code 200 and want to map HTTP status code 429 that is returned by API Gateway to HTTP status code 200.

- For a system error response, the values that are obtained from locations such as `StatusCode`, `Header`, and `BodyJsonField` are all `null`. When you define a mapping condition for an error code mapping plug-in, the value that is obtained from the `ErrorCode` location is `OK` for a backend error response.
- The error code of a system error response is specified by the `X-Ca-Error-Code` header and by the `errorCode` parameter in error logs. This value cannot be overwritten by an `error code mapping plug-in`.
- The `statusCode` parameter in error logs records the value of the HTTP status code that is sent from API Gateway to the client. This value can be overwritten by an `error code mapping plug-in`.

4. Configuration examples

4.1 Use the error codes in error responses for a mapping

Mapping

```
---
# The parameters that are involved in a mapping.
parameters:
 statusCode: "StatusCode"
 resultCode: "BodyJsonField:$.result_code"
 resultId: "BodyJsonField:$.req_msg_id"
# The mapping condition.
errorCondition: "$statusCode = 200 and $resultCode <> 'OK'"
# The parameter in an error response that is used to specify the error code and hit mapping rules.
errorCode: "resultCode"
# Mapping rules.
mappings:
 - code: "ROLE_NOT_EXISTS"
   statusCode: 404
   errorMessage: "Role Not Exists, RequestId=${resultId}"
 - code: "INVALID_PARAMETER"
   statusCode: 400
   errorMessage: "Invalid Parameter, RequestId=${resultId}"
# Optional. The default mapping rule.
defaultMapping:
 statusCode: 500
 errorMessage: "Unknown Error, ${resultCode}, RequestId=${resultId}"
```

5. Limits

- A maximum of 16 parameters can be specified in an error code mapping plug-in.
- A single conditional expression can contain a maximum of 512 characters.

- If you use the `BodyJsonField` location to obtain the JSON string in the body of an error response, the size of the response body cannot exceed **16,380** bytes. If the size of the response body exceeds this limit, the obtained string is null.
- The metadata of an error code mapping plug-in can contain a maximum of `16,380` characters.
- For an error code mapping plug-in, you can configure a maximum of **20** mapping rules by using the `condition` parameter defined in `mappings`.

1.5.4.3. Bind a plugin to an API

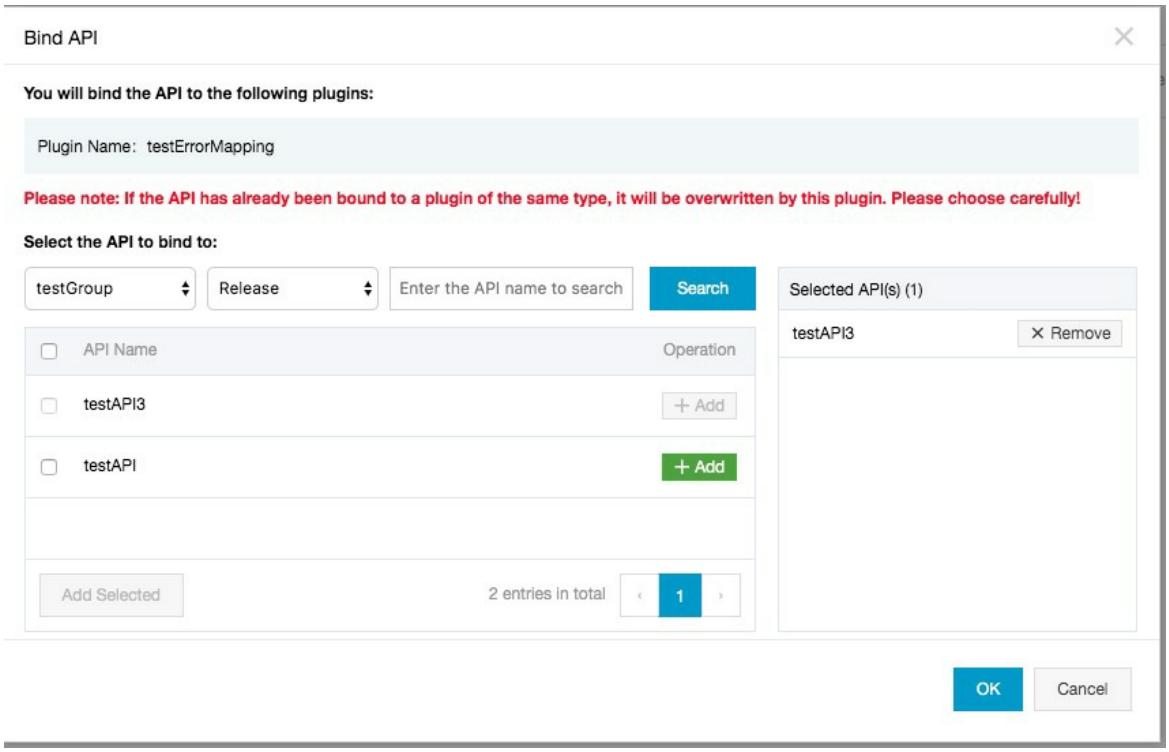
After you create a plugin, you must bind the plugin to an API for the plugin to take effect.

Context

You can bind a plugin to multiple APIs. The plugin will individually take effect on each API. For each type of plugin, you can bind only one plugin of such type to an API. If you bind two plugins of the same type to an API, the new plugin will replace the previous one and take effect.

Procedure

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs** -> **Plugin**.
3. On the Plugins list page, find the target plugin and click **Bind API** in the Operation column.
4. Select the publish environment and the group of the APIs to which you want to bind a plugin.
5. To bind a plugin to one API, find the target API and click +Add in the Operation column to add the API to the right pane. To bind a plugin to multiple APIs, select the target APIs and click Add Selected in the lower-left corner to add these APIs to the right pane. Then, click **OK**.



1.5.4.4. Delete a plugin

You can delete existing plugins.

Procedure

1. Log on to the API Gateway console .
2. In the left-side navigation pane, choose Publish APIs **Plugin**.
3. On the Plugins list page, find the target plugin and click **Delete** in the Operation column.
4. In the Confirm Deletion message, click **OK**.

1.5.4.5. Unbind a plugin

You can unbind plugins from the APIs to which they are bound.

Procedure

1. Log on to the API Gateway console .
2. In the left-side navigation pane, choose Publish APIs **Plugin**.
3. On the Plugins list page, click the name of the target plugin to go to the **Create Plugin** page.
4. Click Bound API List. The bound APIs are displayed. Find the target APIs one at a time and click **Unbind** in the Operation column.
5. In the Confirm Unbind message, click **OK**.

1.6. Manage monitoring

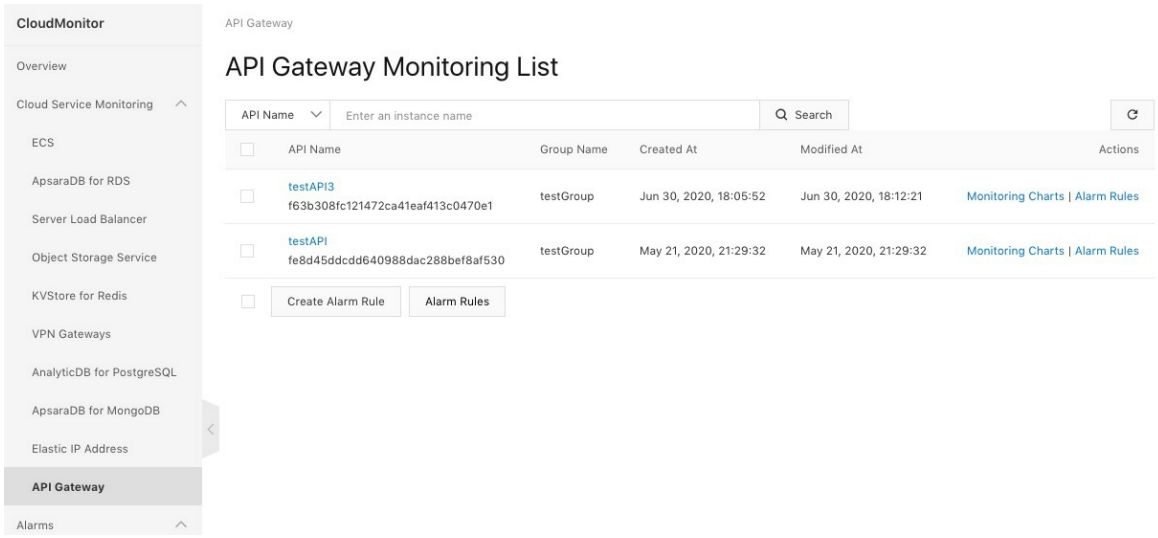
1.6.1. Use CloudMonitor to view monitoring information and configure alert rules

API Gateway works with CloudMonitor to provide visualized real-time monitoring and alerting features. You can use these features to obtain statistical data about your APIs in multiple dimensions, such as the number of API calls, traffic, backend response time, and error distribution. You can view data in different units of time.

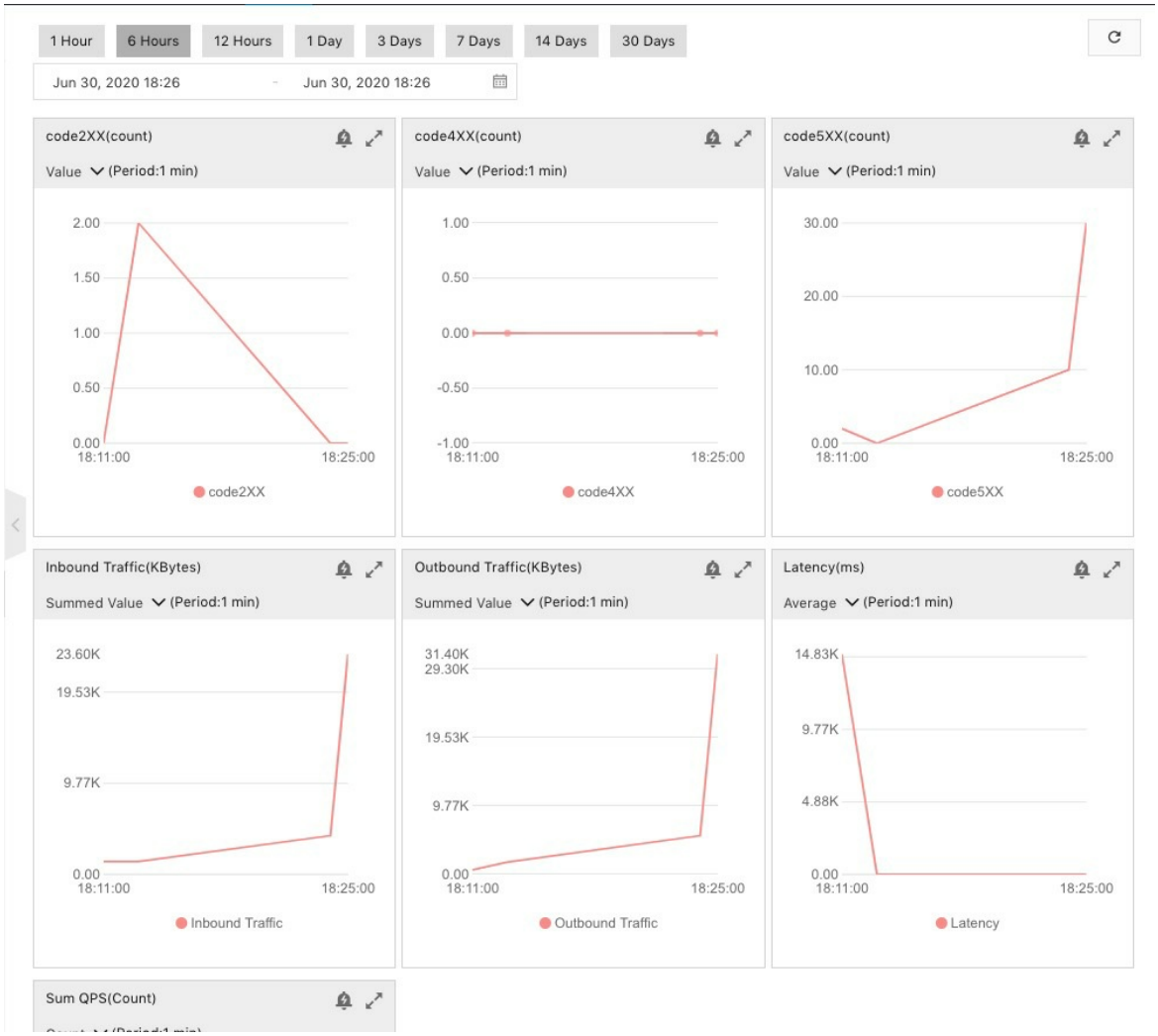
View monitoring information of API calls

Perform the following steps to view the data of API calls within your Apsara Stack tenant account in the CloudMonitor console.

1. In the top navigation bar of the Apsara Uni-manager Management Console, choose **Products > Monitoring and O&M > Cloud Monitor**.
2. In the CloudMonitor console, choose Cloud Service Monitoring > **API Gateway** in the left-side navigation pane.



3. On the API Gateway Monitoring List page, find your API and click **Monitoring Charts** in the Actions column.



Descriptions of monitoring charts:

- code2XX(count)

Shows the number of requests with a 2XX HTTP status code returned. A 2XX HTTP status code, such as 200, indicates that the request succeeded at the backend.

- code4XX(count)

  Shows the number of requests with a 4XX HTTP status code returned. A 4XX HTTP status code, such as 404, indicates a client error.

- code5XX(count)

  Shows the number of requests with a 5XX HTTP status code returned. A 5XX HTTP status code, such as 500, indicates a server error.

- Inbound Traffic(KBytes)

  Shows the size of API requests received.

- Outbound Traffic(KBytes)

  Shows the size of API responses sent.

- Latency(ms)

  Shows the response time of your backend service. The latency in API Gateway ranges from 3 ms to 5 ms, which is excluded from the response time.

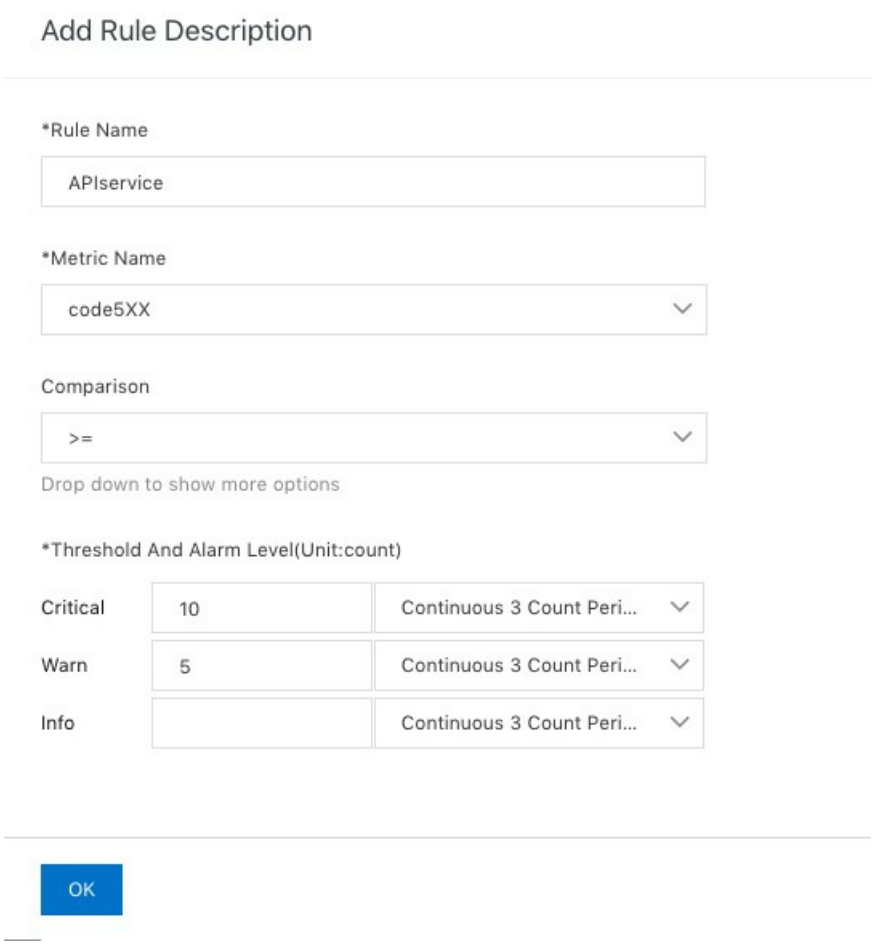
- Sum QPS(Count)

  Shows the total number of API requests.

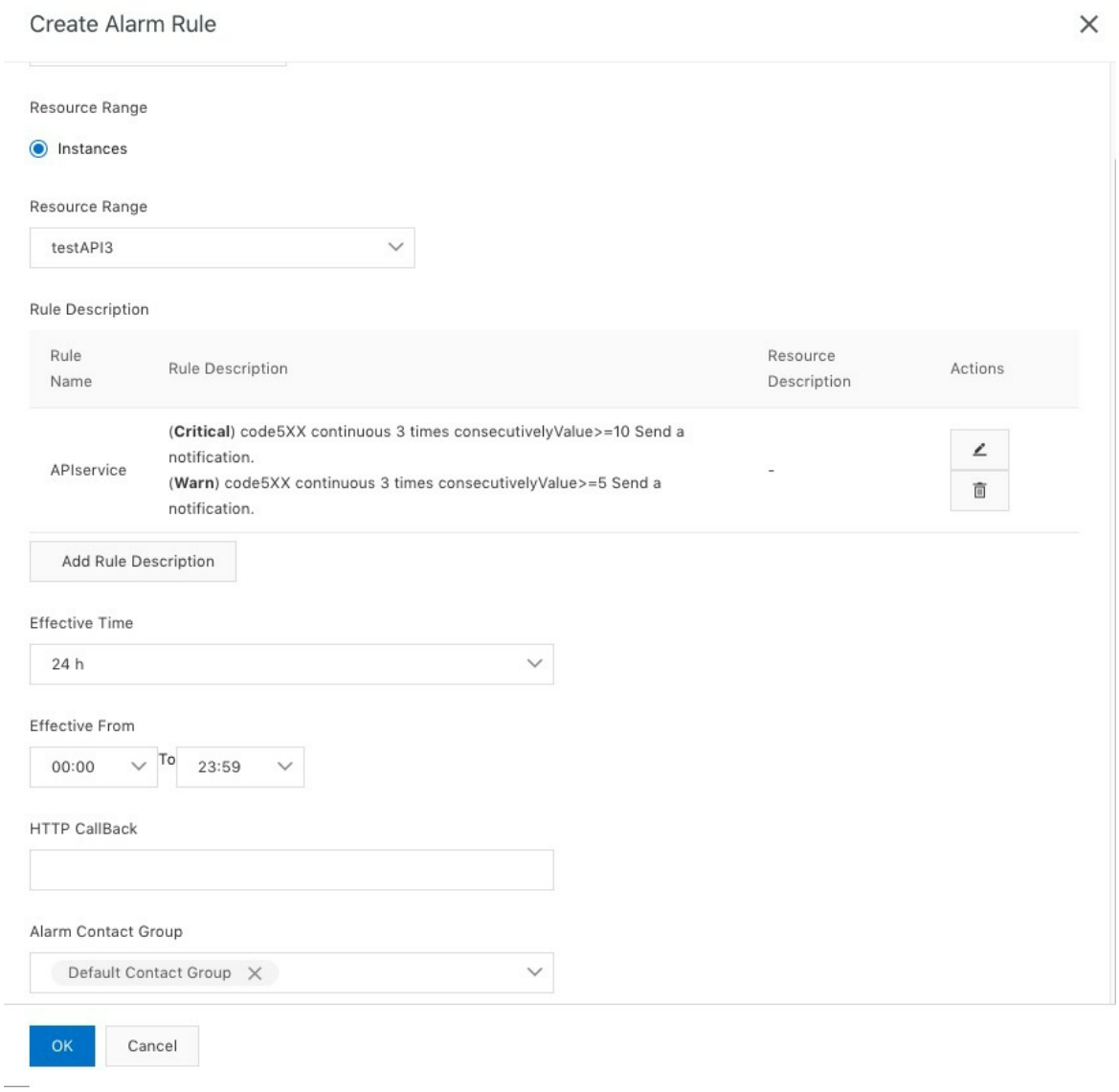
Configure API alert rules

You can configure API alert rules in the CloudMonitor console to achieve real-time API alerting.

1. On the API Gateway Monitoring List page, find your API.
2. Click **Alert Rules** in the Actions column.
3. On the Alert Rules page, click **Create Alert Rule** in the upper-right corner. In the Modify Alert Rule panel, set Product to API Gateway and Resource Range to the required API.
4. Click **Add Rule Description** and specify Rule Name, Metric Name, Comparison, and Threshold And Alert Level. Then, click **OK**.



5. Specify **Alert Contact Group** and click OK.



Note

To monitor the service status of APIs, we recommend that you monitor code5XX(count).

1.6.2. View statistical information on the global monitoring page in API Gateway

You can view monitoring information about API calls in the CloudMonitor console. The API Gateway console also provides an overview page for you to view the statistics of API calls. You can view the statistical information about API calls on the global monitoring page.

> **Notice**
>
> On the global monitoring page, you can view only the information about API calls in a specific API group or the calling information about a specific API within your account. Even user root cannot view all the data on the global monitoring page.

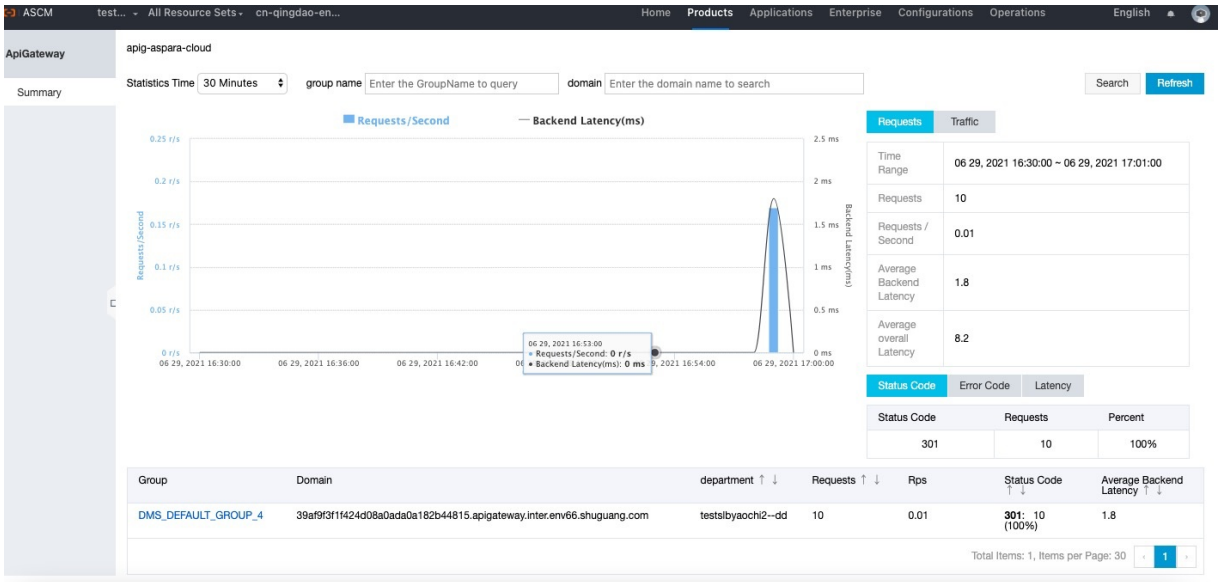
View overall monitoring information

1. Log on to the API Gateway console. In the left-side navigation pane, click **Instances**. On the Instance list page, find the desired instance and click **Global Monitoring** in the upper-right corner. The summary page appears.
2. Specify Statistics Time and specify group name or domain to view specific information about API calls within your account. By default, the Statistics Time parameter is set to 30 Minutes.

You can view the following metrics:

- Requests: On this tab, you can view the following metrics within your account: Requests, Requests / Second, Average Backend Latency, and Average overall Latency.
- Traffic: On this tab, you can view the following metrics within your account: Request Traffic, Response Traffic, Average Request Size, and Average Response Size.
- Status Code: On this tab, you can view the returned status codes within your account, and the number and percentage of requests with each returned status code.
- Error Code: On this tab, you can view the returned error codes within your account, and the number and percentage of requests with each returned error code.
- Latency: On this tab, you can view the number and percentage of requests within a specific latency range.

In addition, you can view the domain name bound to each API group, and can sort API groups by department, number of requests, requests per second (RPS), status code, or average backend latency.



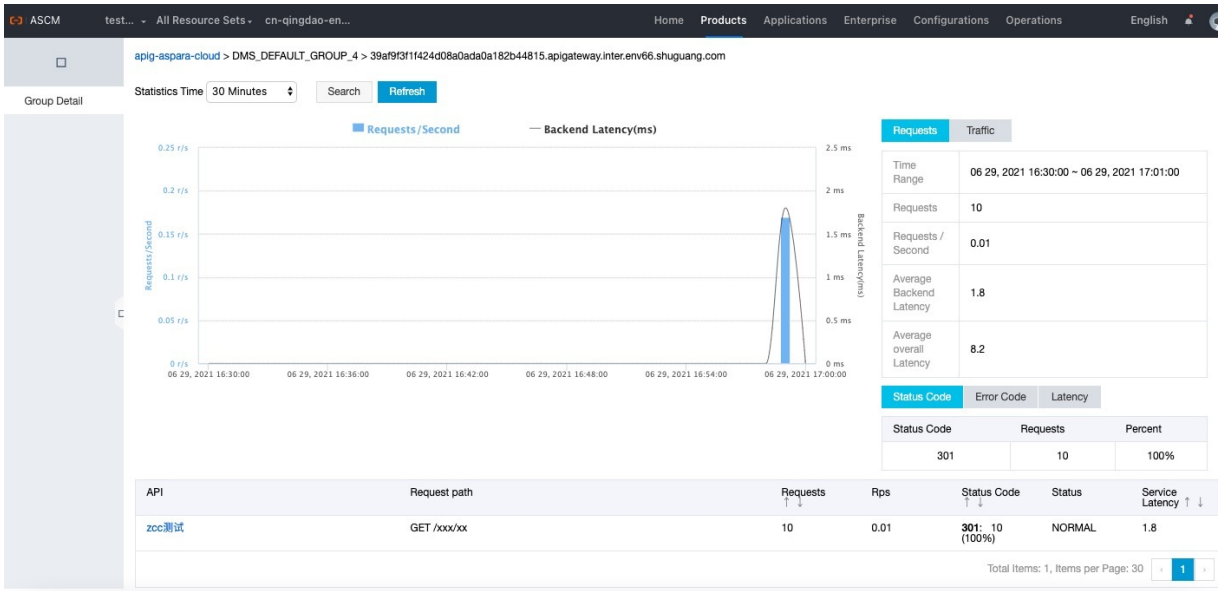
View monitoring information of a specific API group

On the Summary page, click the name of the desired API group. On the Group Detail page, specify Statistics Time to view specific information about API calls in this API group. By default, the Statistics Time parameter is set to 30 Minutes.

You can view the following metrics:

- Requests: On this tab, you can view the following metrics within your API group: Requests, Requests / Second, Average Backend Latency, and Average overall Latency.
- Traffic: On this tab, you can view the following metrics within your API group: Request Traffic, Response Traffic, Average Request Size, and Average Response Size.
- Status Code: On this tab, you can view the returned status codes within a specified period of time in your API group, and the number and percentage of requests with each returned status code.
- Error Code: On this tab, you can view the returned error codes within a specified period of time in your API group, and the number and percentage of requests with each returned error code.
- Latency: On this tab, you can view the number and percentage of requests within a specific latency range in your API group.

In addition, you can view the request paths of all APIs in your API group, and can sort APIs by number of requests, RPS, status code, or latency.



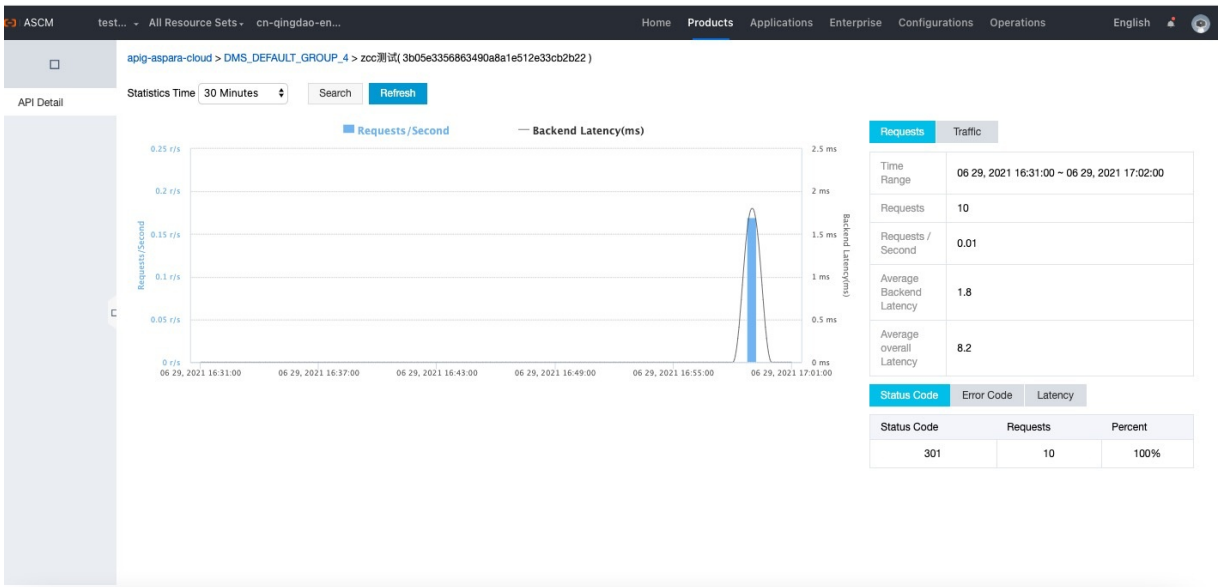
View monitoring information of a specific API

On the Group Detail page, click the name of the desired API. On the API Detail page, specify Statistics Time to view the specific calling information about the API. By default, the Statistics Time parameter is set to 30 Minutes.

You can view the following metrics:

- Requests: On this tab, you can view the following metrics: Requests, Requests / Second, Average Backend Latency, and Average overall Latency.
- Traffic: On this tab, you can view the following metrics: Request Traffic, Response Traffic, Average Request Size, and Average Response Size.
- Status Code: On this tab, you can view the returned status codes within a specified period of time, and the number and percentage of requests with each returned status code.

- Error Code: On this tab, you can view the returned error codes within a specified period of time, and the number and percentage of requests with each returned error code.
- Latency: On this tab, you can view the number and percentage of requests within a specific latency range.



1.6.3. Configure an account for global monitoring

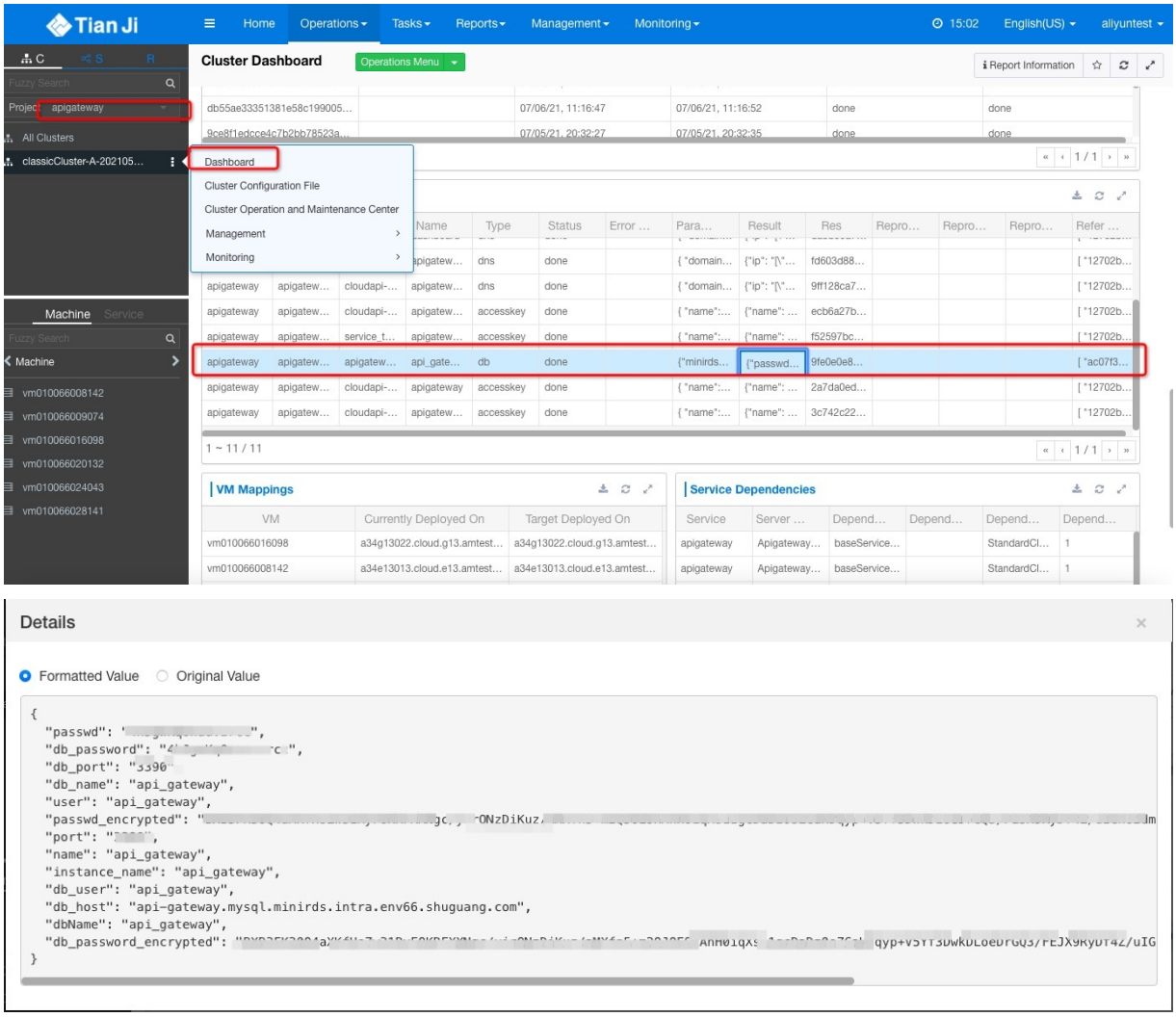
API Gateway provides the global monitoring feature. You can configure a global administrator account that can be used to view the API calls in all departments.

Configure global monitoring

To configure global monitoring, you must configure a global administrator account first. To configure a global administrator account, you must add the PrimaryKey value of the Level-1 department to which the global administrator account belongs to the desired API Gateway database.

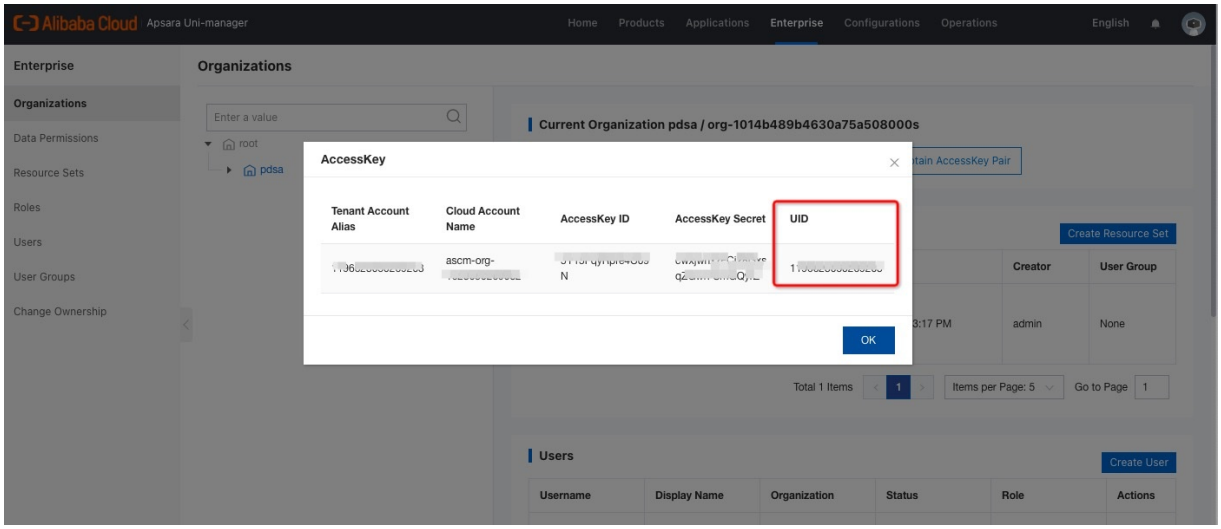
Step 1: Obtain the username and password that are used to access the desired database.

Log on to the Apsara Infrastructure Management Framework console. In the left-side navigation pane, choose Tools > Operation Tools > Machine Tools. The Operation Tools page appears. On the C tab in the left-side navigation pane, select apigateway from the Project drop-down list. Move the pointer over the More icon next to one of the listed clusters and select Dashboard from the shortcut menu. In the Cluster Resource section, find the cluster for which the Type parameter is set to db. Then, right-click the value in the Result column and select Show More from the shortcut menu to view the details of the desired database. The db_user parameter specifies the username that is used to access the database. The db_host parameter specifies the address of the database. The db_password parameter specifies the password that is used to access the database. The db_port parameter specifies the port of the database. The db_name parameter specifies the name of the database.



Step 2: Obtain the PrimaryKey value of the Level-1 department to which the global administrator account belongs.

Log on to the Apsara Uni-manager Management Console. In the top navigation bar, click Enterprise. Select the Level-1 department to which the global administrator account belongs in the left-side navigation pane of the Organizations page and click Obtain AccessKey Pair. In the AccessKey message, view the PrimaryKey value.



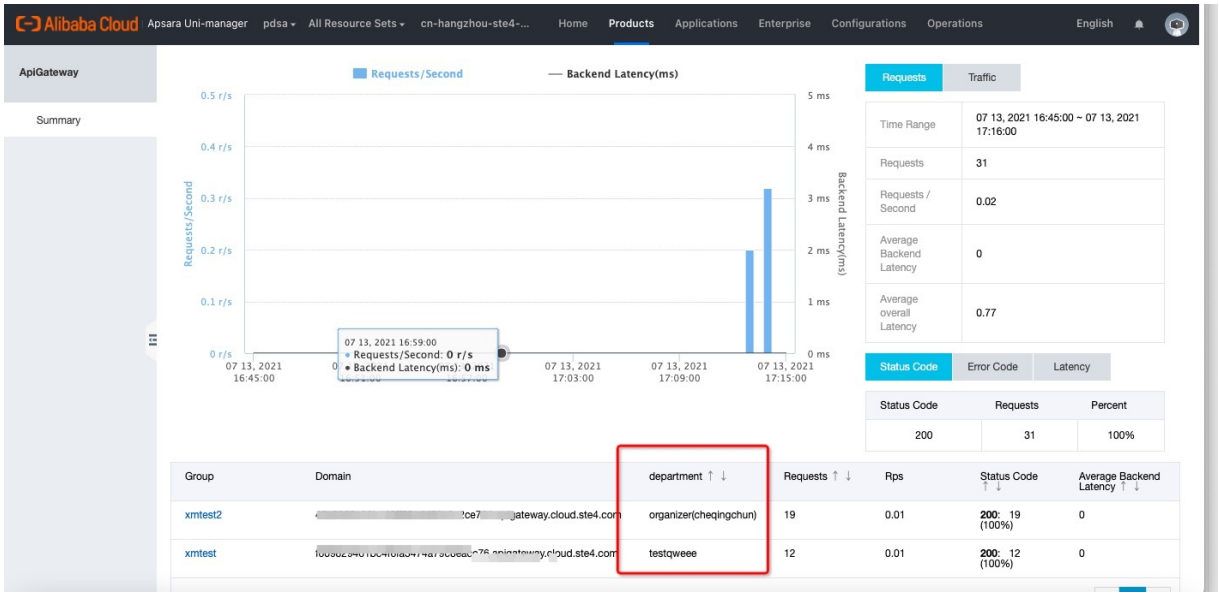
Step 3: Log on to the database to configure the global administrator account.

In the Apsara Infrastructure Management Framework console, log on to the host for which the serverRole parameter is set to apigateway.ApigatewayOpenAPIForClassic. Then, run the following commands to access the desired database and configure the global administrator account based on the information obtained in Step 1 and Step 2.

```
mysql -u api_gateway -hapi-gateway.mysql.minirds.intra.env66.shuguang.com -pemSJzawsl70ybhll -P 3157    # Log on to the database environment.
use api_gateway;   # Select the api_gateway database.
INSERT INTO `ca_stastic_config`(`gmt_create`,`gmt_modified`,`config_key`,`config_value`,`is_removed`) VALUES(NOW(),NOW(),'AdminUidConfig','["xxx"]', 0);      # Replace xxx with the PrimaryKey value that serves as the uid value to configure the global administrator account. You can configure multiple uid values.
UPDATE `ca_stastic_config` SET `config_value` = '["xxx"]' WHERE `config_key` = 'AdminUidConfig';
# Update configurations.
```

Step 4: Use the global administrator account to view monitoring details.

Log on to the API Gateway console. Click Instances in the left-side navigation pane. On the Instance list page, find the desired instance and click Global Monitoring. On the Summary page, you can filter data by API group or domain name based on the specified period of time.



Note: By default, only the monitoring data of API groups in the Level-1 department can be displayed. Only the configured global administrator account can be used to view monitoring data in all departments.

1.7. Advanced usage

1.7.1. Customize business parameters for logs

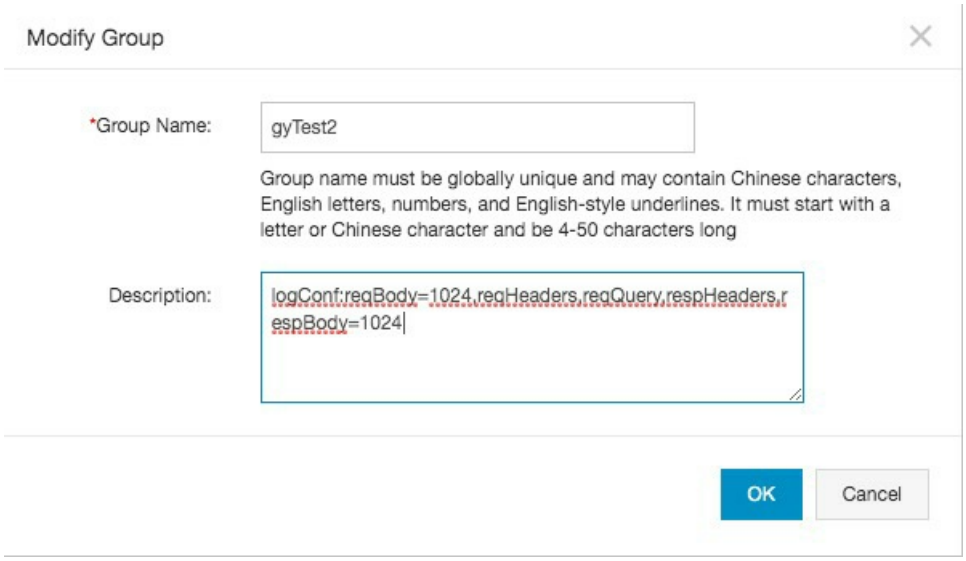
API Gateway provides the Hack mode, which allows you to record the request and response parameters of API calls in logs.

Procedure

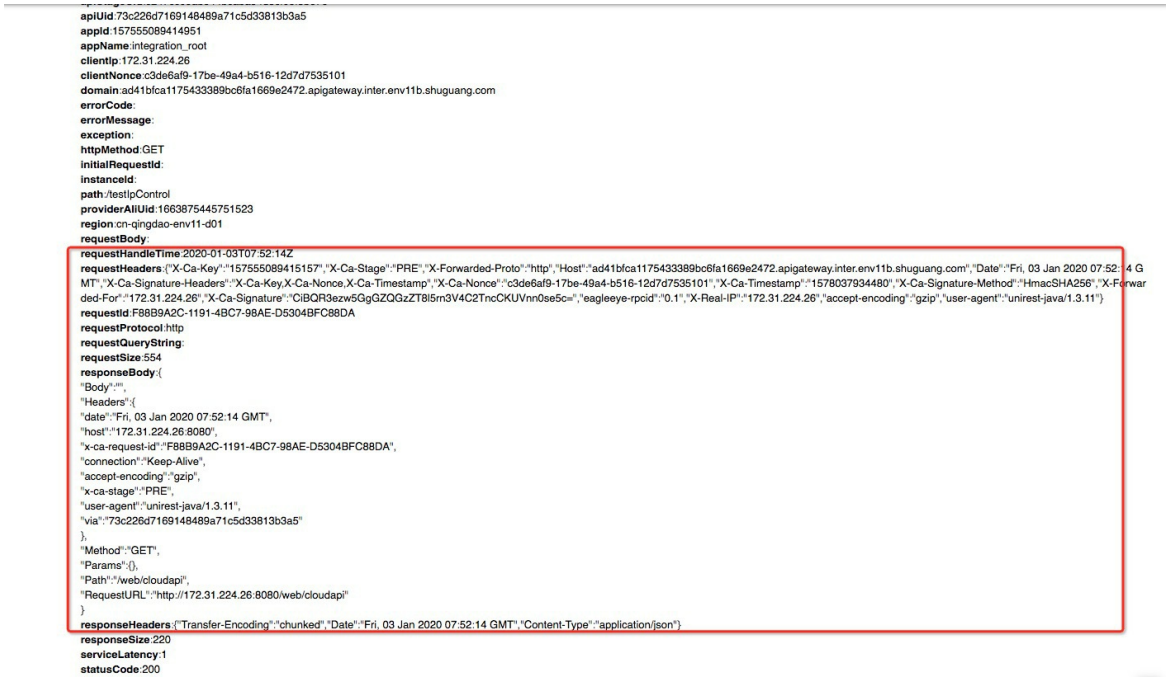
1. Log on to the API Gateway console
2. In the left-side navigation pane, choose **Publish APIs > API Groups**.
3. On the Group List page, find your API group and click its name to go to the Group Details page.
4. Click **Modify Group Message** in the upper-right corner.
5. In the **Description** field of the Modify Group dialog box, add the following content:

```
logConf:reqBody=1024,reqHeaders,reqQuery,respHeaders,respBody=1024
```

> **Note**
>
> - The content must be in a separate line. Otherwise, the configuration fails.
> - You can also modify the content based on your requirements. For example, you can remove respHeaders to omit the response header and its content in the logs.
> - reqBody=1024 indicates that the log records a maximum of 1,024 characters from the request body. Extra characters are discarded.



6. Log on to the Log Service console and check whether the description change has been recorded.



1.7.2. Configure Log Service logs for API Gateway

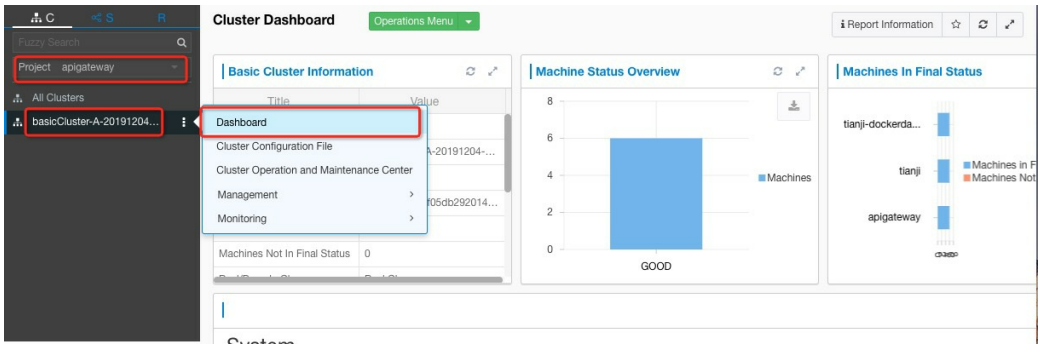
1.7.2.1. Initialize the default Log Service configuration of API Gateway

By default, API call logs of API Gateway are synchronized to Log Service in Apsara Infrastructure Management Framework. However, your account can be activated only after you log on to the Log Service console from Apsara Infrastructure Management Framework.

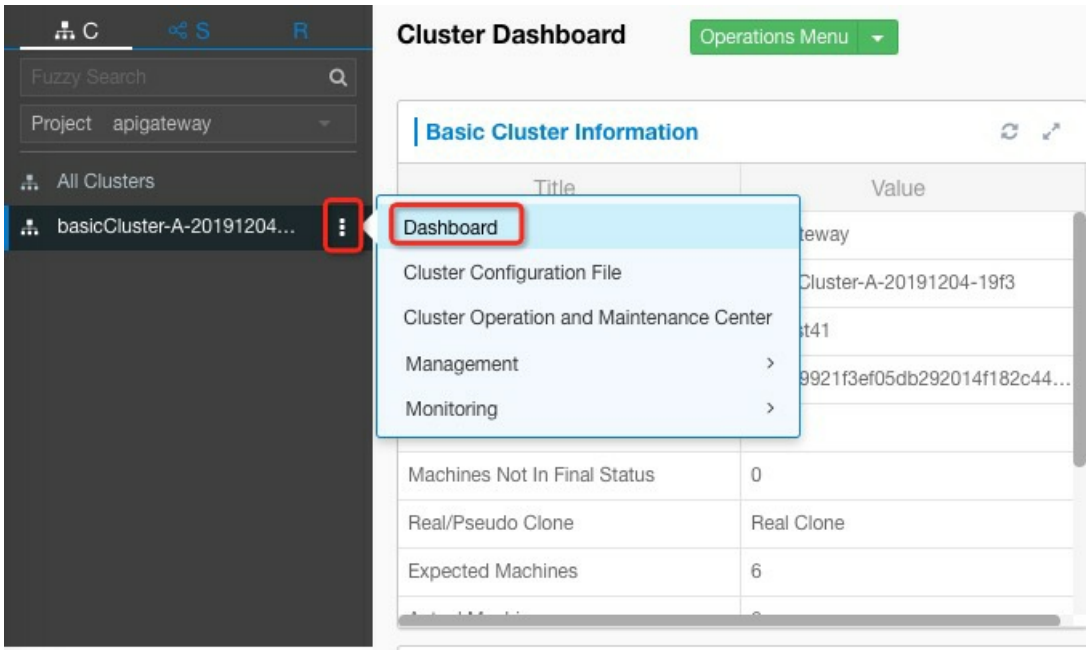
Procedure

1. Log on to the Apsara Infrastructure Management Framework console. In the left-side navigation pane, choose Tools > Operation Tools > Machine Tools. The Operation Tools page appears. You can also click `Go` on the Machine Tools page to go to the Operation Tools page again.
2. Click the **C** tab in the left-side navigation pane. Then, select *apigateway* from the **Project** drop-down list.

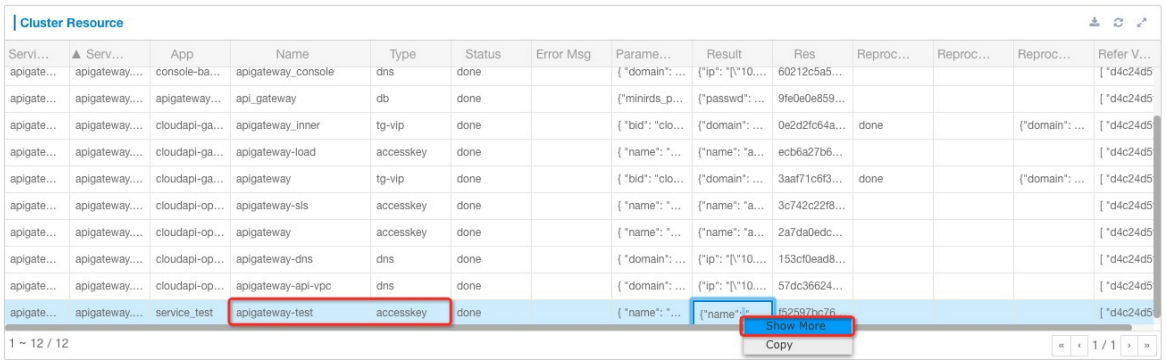
   Cluster O&M page

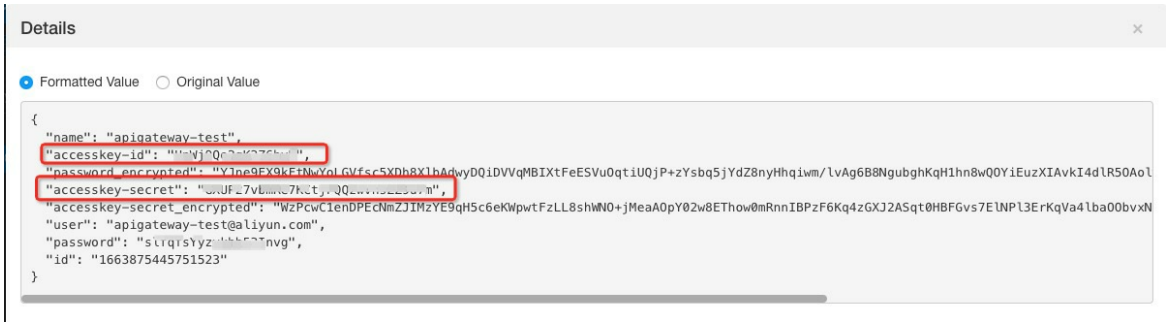


3. Move the pointer over the icon next to one of the filtered clusters, and select **Dashboard** from the shortcut menu.



4. In the Cluster Resource section of the page that appears, find the service with Name set to apigateway-sls and Type set to accesskey. Right-click the value in the Result column and select **Show More** from the shortcut menu to view the values of accesskey-id and accesskey-secret.

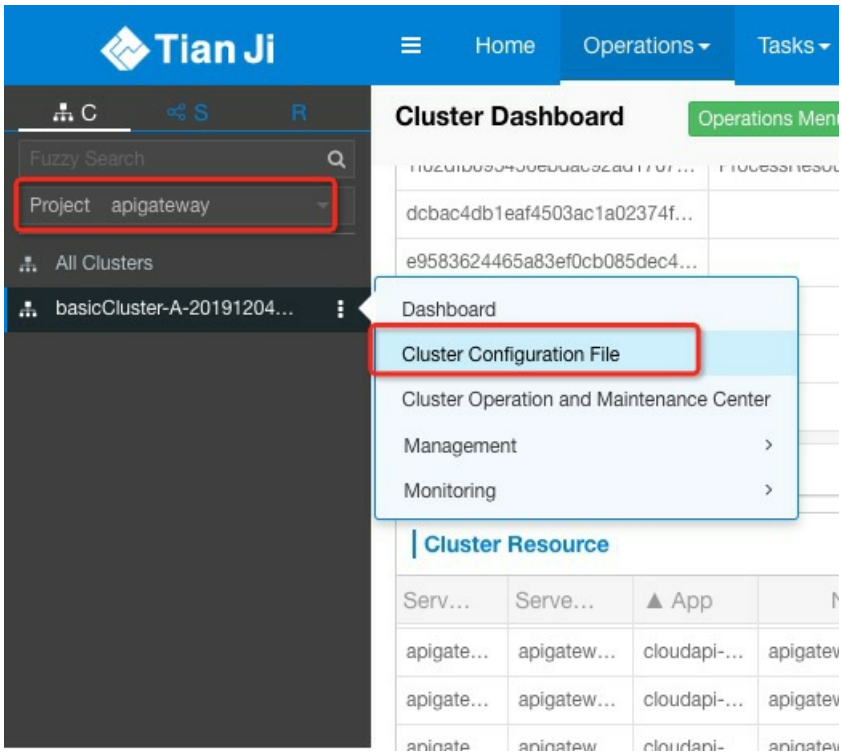




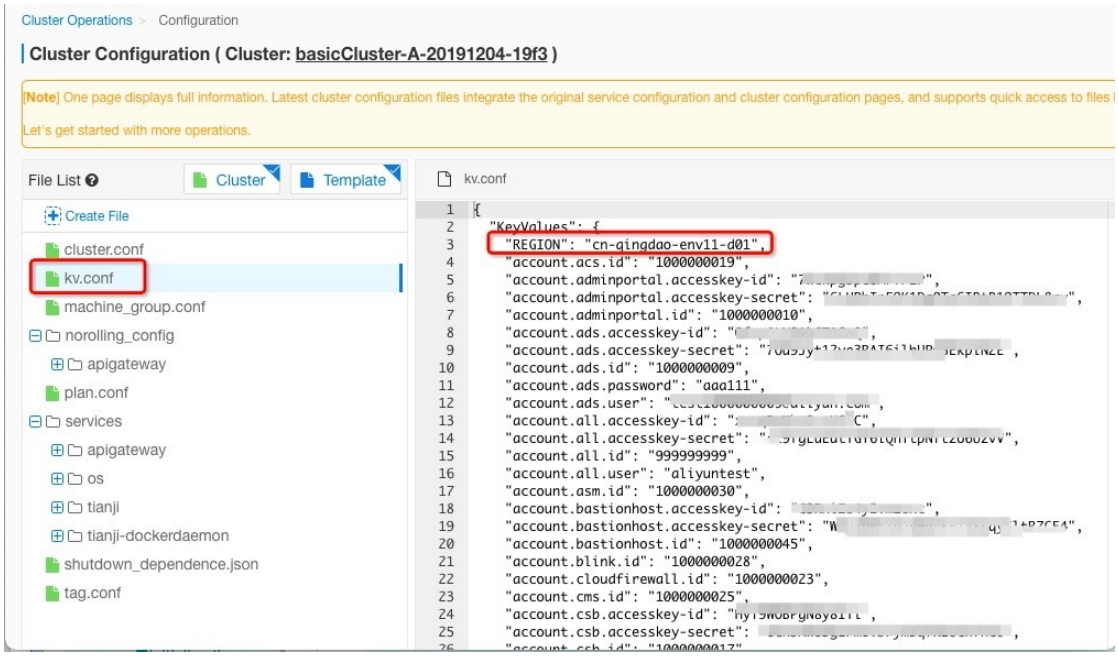
5. Use the accesskey-id and accesskey-secret values that you obtained to log on to the Log Service console from Apsara Infrastructure Management Framework.

   The URL for the Log Service console in Apsara Infrastructure Management Framework is in the following format: http://portal.${region}.sls.${internet-domain}. You can obtain the values of region and internet-domain from the kv.conf file in the Apsara Infrastructure Management Framework console.

   i. Move the pointer over the More icon next to the apigateway cluster and select **Cluster Configuration File** from the shortcut menu.



ii. Click the **kv.conf** file to view the values of region and internet-domain.



Note After you log on to the Log Service console, Log Service is automatically configured for API Gateway. This operation takes several minutes.

1.7.2.2. Configure API Gateway to ship logs to your Log Service project

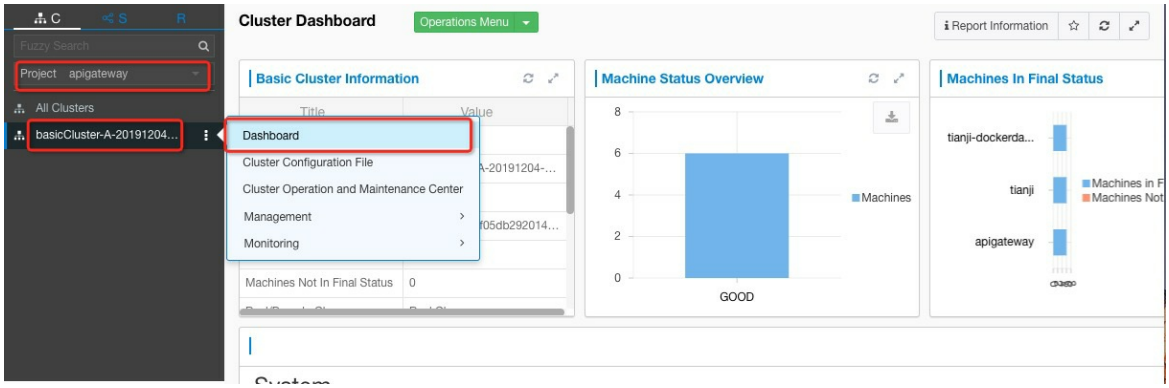
If you do not want to use Log Service and a fixed account in Apsara Infrastructure Management Framework, you can create a Log Service project. Then, you can configure API Gateway to ship logs to your Log Service project.

Context

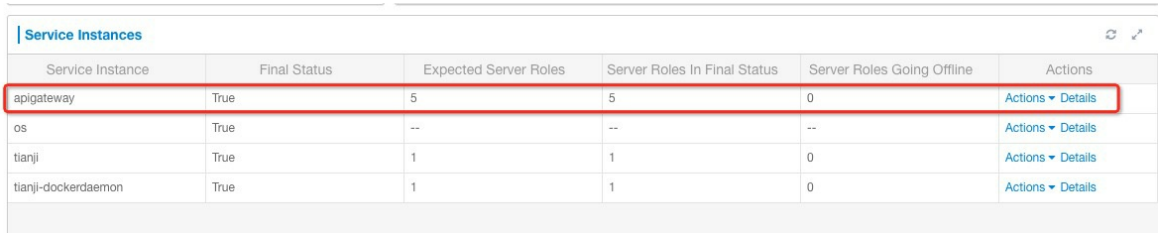
You must perform the following steps to create a Log Service project and configure Logstores and machine groups:

1. Log on to the machine where API Gateway resides

1. Log on to the Apsara Infrastructure Management Framework console. In the left-side navigation pane, choose Tools > Operation Tools > Machine Tools. The Operation Tools page appears. Click the C tab in the left-side navigation pane. Then, select apigateway from the Project drop-down list. Move the pointer over the More icon next to one of the filtered clusters, and select Dashboard from the shortcut menu.



2. In the **Service Instances** section of the page that appears, find the row where the value of Service Instance is apigateway.



Service Instances

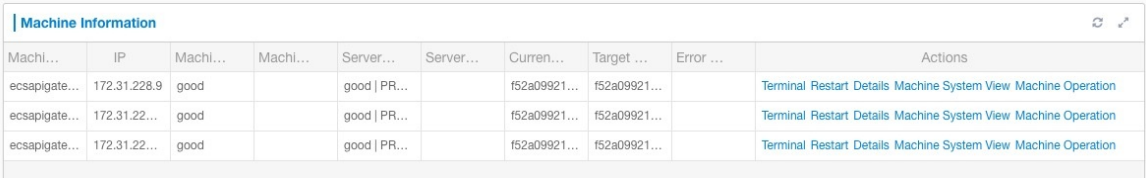
Service Instance	Final Status	Expected Server Roles	Server Roles In Final Status	Server Roles Going Offline	Actions
apigateway	True	5	5	0	Actions Details
os	True	--	--	--	Actions Details
tianji	True	1	1	0	Actions Details
tianji-dockerdaemon	True	1	1	0	Actions Details

3. Click **Details** in the Actions column. In the **Server Role List** section of the page that appears, find the row where the value of Server Role is ApigatewayLite#.

Server Role List

Server Role	Current Status	Expected Machines	Machines In Final ...	Machines Going ...	Rolling Task Status	Time Used	Actions
ApigatewayConsole#	In Final Status	2	2	0	no rolling		Details
ApigatewayDB#	In Final Status	1	1	0	no rolling		Details
ApigatewayLite#	In Final Status	3	3	0	no rolling		Details
ApigatewayOpenAPI#	In Final Status	2	2	0	no rolling		Details
ServiceTest#	In Final Status	1	1	0	no rolling		Details

4. Click **Details** in the Actions column. Find the required machine in the **Machine Information** section and click Terminal in the Actions column to log on to the machine.

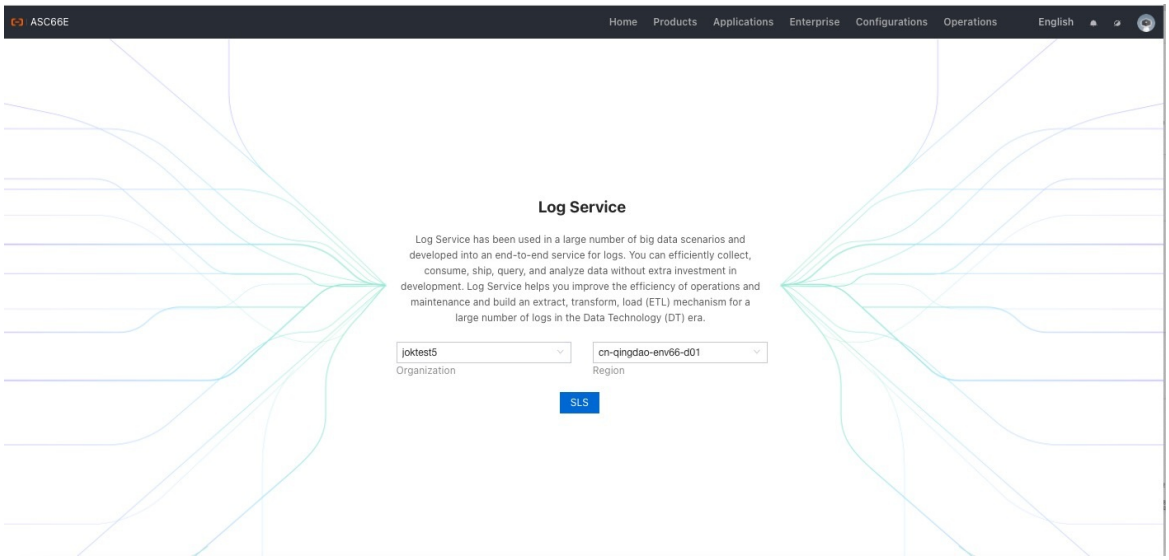


Machine Information

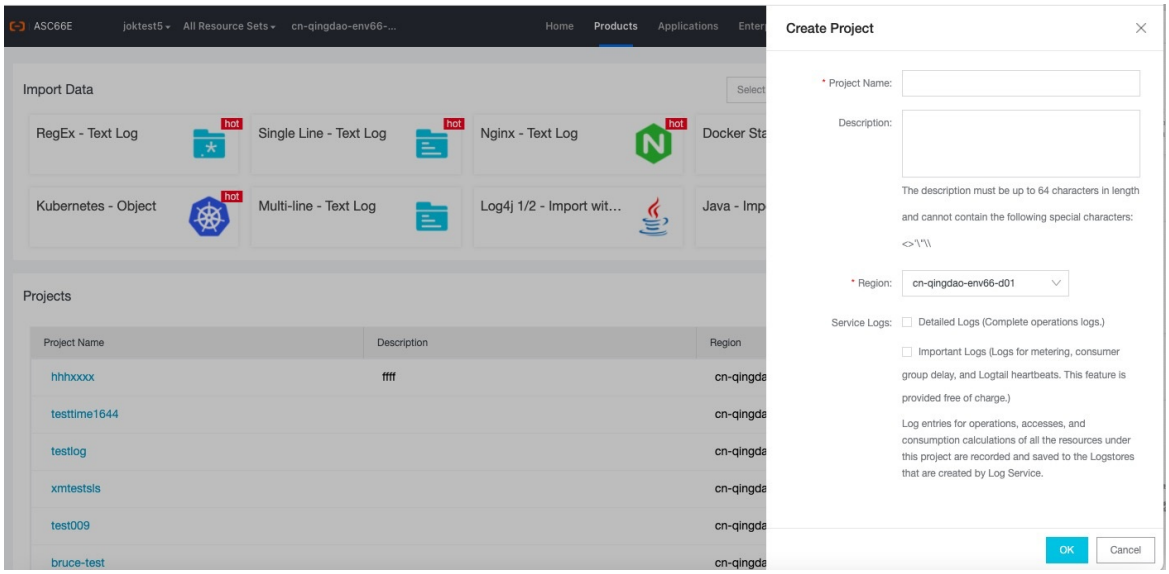
Machi...	IP	Machi...	Machi...	Server...	Server...	Curren...	Target ...	Error ...	Actions
ecsapigate...	172.31.228.9	good		good | PR...		f52a09921...	f52a09921...		Terminal Restart Details Machine System View Machine Operation
ecsapigate...	172.31.22...	good		good | PR...		f52a09921...	f52a09921...		Terminal Restart Details Machine System View Machine Operation
ecsapigate...	172.31.22...	good		good | PR...		f52a09921...	f52a09921...		Terminal Restart Details Machine System View Machine Operation

2. Configure logs in the Log Service console

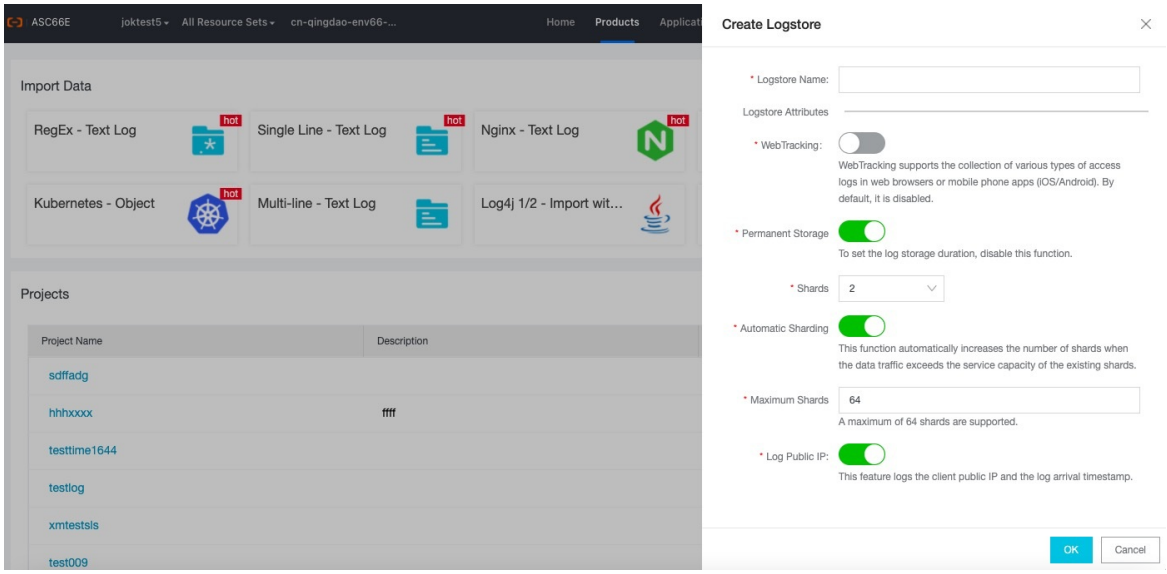
1. Log on to the Apsara Uni-manager Management Console. In the top navigation bar, choose Products —— Application Services —— Log Service. On the page that appears, specify **Region** and **Organization** and click **SLS**.



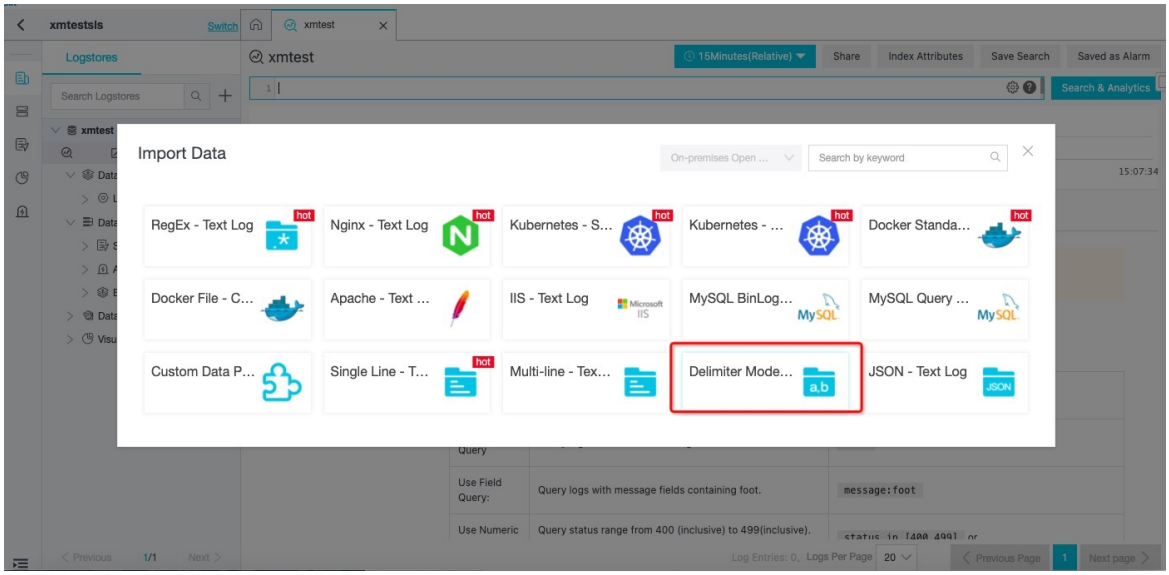
2. Click Create Project. In the Create Project panel, configure parameters and click OK.



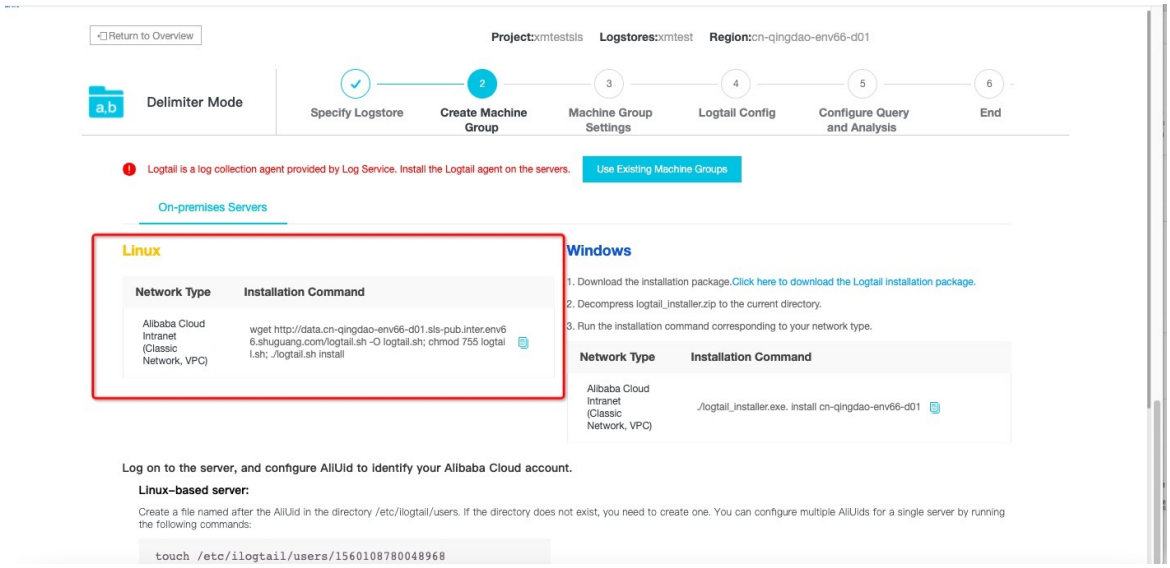
3. In the Created message, click OK. In the Create Logstore panel, configure parameters and click OK to create a Logstore.



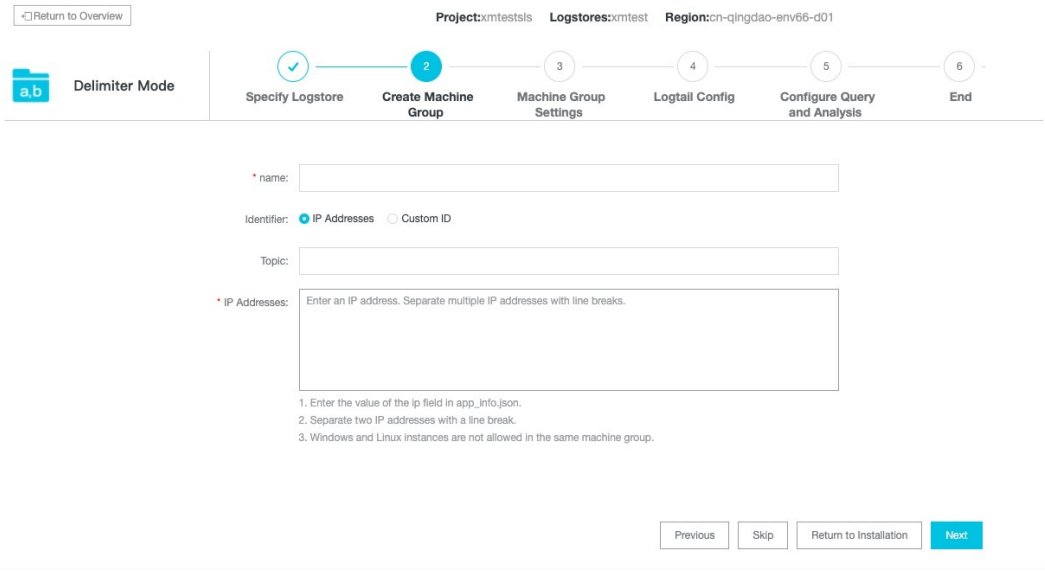
4. In the Created message, click OK. On the Import Data page, click **Delimiter Mode - Text Log**.



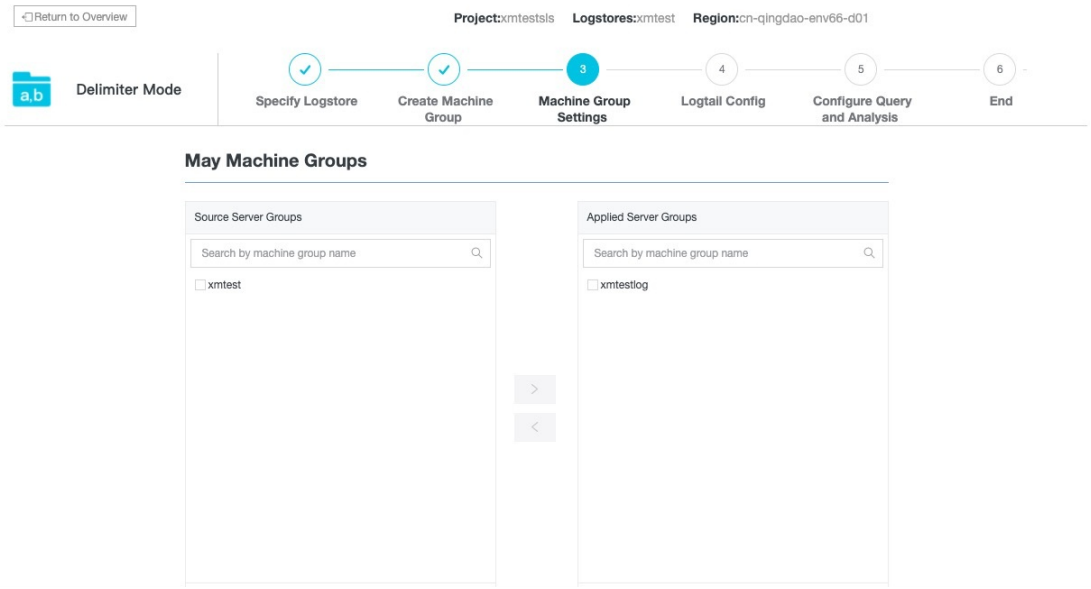
5. Access the container whose service role starts with ApigatewayLite in the Apsara Infrastructure Management Framework console. Then, run the installation command that is displayed in the Create Machine step on the TerminalService page. The operations are similar to those in the first section. If the "install logtail success" message appears, the installation succeeds.



6. After the installation is complete, click Complete Installation and configure parameters in the Create Machine step. Specify name. Set IP Addresses to the IP addresses of all the Docker containers whose service role starts with ApigatewayLite.



7. Click Next to go to the Machine Group step. Configure machine groups and click Next. A heartbeat check automatically starts. If the check succeeds, you are redirected to the Logtail Config step.

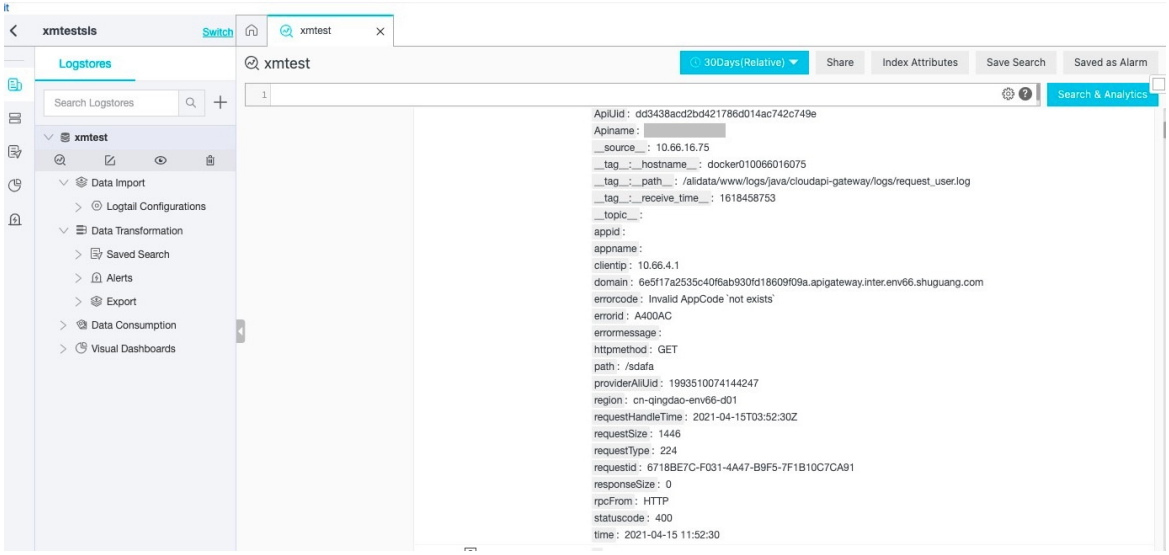


8. In the Logtail Config step, configure the parameters that are described in the following table and click **Next**.

Parameter	Value
Config Name	gateway_request_log
Log Path	/alidata/www/logs/java/cloudapi-gateway/logs/request_user.log
Mode	Delimiter Mode
Log Sample	2019-08-15 14:25:05|CC7C526B-C915-44F1-93A2-F44DBBE35177|b2909c9fa66146f19baf2bd8f4709ab8|integration_root|e4032f87ace14cc6965cf5352a9637a6|RELEASE|0619f7763b004fc3a16a41dd712ce8d7|biz1_anonymous|10.4.21.241|||b2909c9fa66146f19baf2bd8f4709ab8.apigateway.env4b.shuguang.com|POST|/biz1/anonymous|403|A403JT:Invalid JWT: deserialized JWT failed||1453964555641148|cn-qingdao-env4b-d01|2019-08-15T06:25:05Z|614|0|0|A403JT|http|||cf802da1-54d0-49b8-b77c-2b20b172d90a||{"X-JWT-Token":"bad jwt token"}|a=%21111||||||

Parameter	Value
Delimiter	Vertical Line

9. Use the default settings until the entire configuration process is complete.
10. After the configuration is complete, make some API calls in the API Gateway console and check whether the configuration takes effect in the Log Service console.



1.7.3. Cross-user VPC authorization

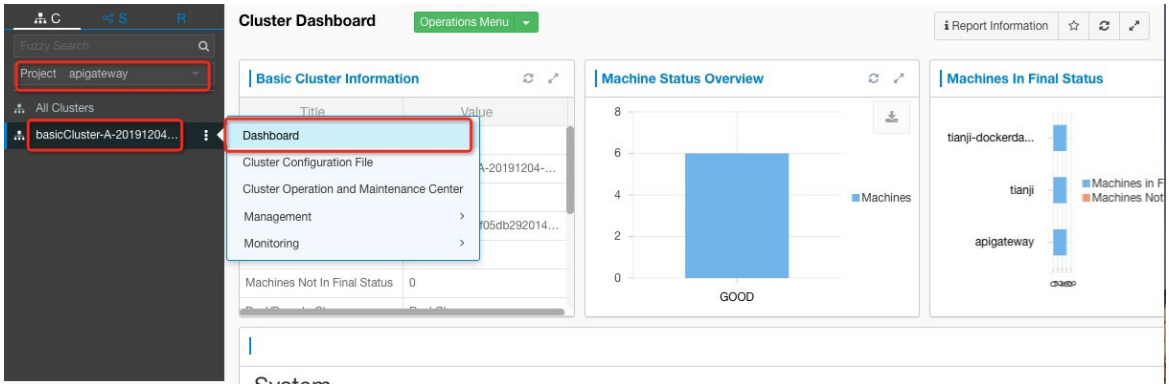
If your backend service resides in a VPC, you must configure the backend service address through VPC authorization. By default, a VPC owner must be the same user as an API owner. Starting from Apsara Stack V3.8.1, API Gateway provides two internal APIs that allow VPC owners to authorize their VPCs to other users.

1.7.3.1. User authorization across VPCs

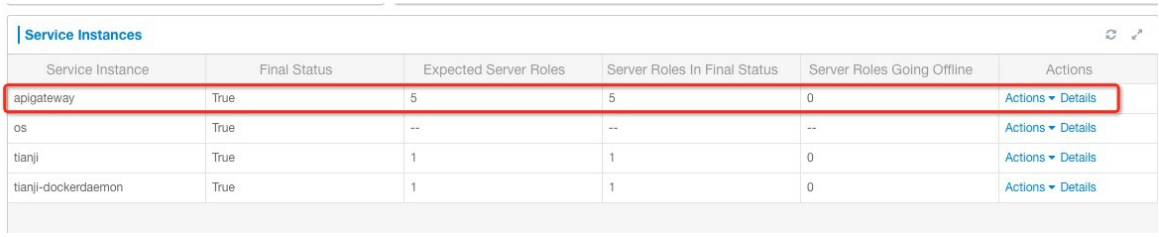
APIs can be used across multiple VPCs. For security reasons, VPC owners must call APIs to explicitly authorize access to VPCs before API providers can use the VPCs. The OpenAPI component of API Gateway has a built-in Aliyun CLI tool. You can use this tool to authorize access to VPCs. The following steps describe how to call the API:

Procedure

1. Log on to the Apsara Infrastructure Management Framework console. In the left-side navigation pane, click the C tab and select apigateway from the Project drop-down list. Place the pointer over the More icon next to one of the filtered clusters and choose Dashboard from the shortcut menu.

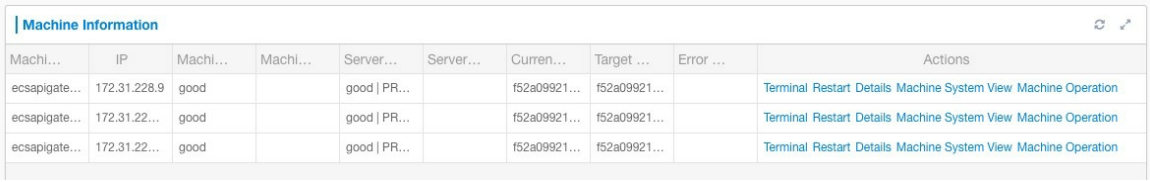


2. On the dashboard page, go to **Service Instance List**, and find apigateway in the Service Instance column.



3. Click **Details** in the Actions column corresponding to the apigateway service instance. In the **Server Role List** section, find ApigatewayLite# in the Server Role column.

4. Click **Details** in the Actions column corresponding to the ApigatewayLite# server role. Then, find the **Machine Information** section.



5. Click Terminal in the Actions column corresponding to a machine in the server role to access the container.
6. Configure the AccessKey pair used to call the CLI tool. Run the following commands:

```
aliyun configure -- profile vpctest //Add the AccessKey pair configuration. vpctest is the profile name, w
hich can be customized. After you press Enter, configure AccessKeyId and AccessSecret as prompted.
aliyun configure list //View the config configuration to check whether the preceding profile has been ad
ded.
```

7. Run the following command to perform authorization:

```
aliyun cloudapi AuthorizeVpc --VpcId vpc-tb5mfcwx3s4zqctzw**** --TargetUserId 147546214349****
--endpoint apigateway.cn-qingdao-env11-d01.inter.env11b.shuguang.com --force
--profile vpctest
```

Note

- Log on to the Apsara Infrastructure Management Framework console. In the top navigation bar, choose **Reports** > **System Reports**. On the System Reports page, click Registration Vars of Services. On the Registration Vars of Services report, right-click the value in the Service Registration column corresponding to the apigateway service and choose Show More from the shortcut menu. The value of the apigateway.openapi.endpoint variable must be used as the endpoint in the preceding command. The profile value also needs to be replaced based on actual needs.
- This command authorizes the user whose ID is 1475462143497330 to use the VPC with the ID vpc-tb5mfcwx3s4zqctzw19w2. You can replace the parameter values as needed.

Success Operation Sample

```
[root@docker010011102023 ~]
#aliyun cloudapi AuthorizeVpc --VpcId vpc-tb5mfcwx3s4zqctzw19w2 --TargetUserId 1475462143497330 --endpoint apigateway.cn-qingdao-env
11-d01.inter.env11b.shuguang.com --force --profile vpctest
{"RequestId":"8E64BC51-60D7-4D85-B5F0-3E3F12C4B6D2"}
```

1.7.3.2. Configure APIs

After an app is authorized to call an API, the API owner must configure the API and define the API backend service. This is because you cannot select the VPC ID of other users in the Apsara Uni-manager Management Console.

Context

When you configure an API, take note of the following points:

- Backend Service Type cannot be set to VPC.
- The backend service address must be in the http(s)://{Backend service IP address}.{vpcId}.gateway.vpc:{port} format. The content in {} can be substituted as required. Example:

  ```
  http://192.168.XX.XX.vpc-tb5mfcwx3s4zqctzw****.gateway.vpc:8080
  ```

Procedure

1. Log on to the API Gateway console
2. In the left-side navigation pane, choose **Publish APIs** > **APIs**.
3. On the API List page, find your API and perform the following operations:
   - Click the name of the API to go to the **API Definition** page. You can view information about the API.
   - Click **Edit** in the upper-right corner, modify configurations as required, and then click **Save**.

     The procedure of creating an API is similar to that of modifying an API. For more information about how to create an API, see Create an API. If you want to cancel the modifications before the modifications are submitted, click Cancel Edit in the upper-right corner of the edit page.

1.7.4. Call an API over HTTPS

Context

API Gateway locates a unique API group by domain name, and locates a unique API in the API group by using Path and HTTPMethod. API Gateway assigns a second-level domain for each API group. You can use the domain name to call APIs that belong to the API group. The second-level domain supports only access over HTTP. You can also use a custom domain name to call APIs. This topic describes how to call APIs by using a second-level domain or by using a custom domain name.

Use a second-level domain to call APIs over HTTPS

To use a second-level domain to call APIs over HTTPS, you must perform the following steps to configure a wildcard domain name certificate in Apsara Stack. You must prepare the certificate yourself.

1. Prepare configuration files for a second-level domain.

   Modify configurations in the following code: Replace *.wildcard.com with your wildcard domain name *.apigateway.${internet-domain}. You can obtain the value of the ${internet-domain} variable in the kv.conf configuration file for Apsara Infrastructure Management Framework. For example, if the domain name that you use to provide external services is abc.alibaba.com, replace *.wildcard.com with *.alibaba.com. Save the modified code to a wildcard.conf file. Set the name of the public key file in the certificate to wildcard.crt. Set the name of the private key file in the certificate to wildcard.key.

   ```
   server {
     listen          443 http2 ssl;
     server_name      *.wildcard.com;
     limit_req       zone=perserver_req burst=100;
     ssl_protocols     TLSv1 TLSv1.1 TLSv1.2;
     ssl_ciphers      ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-SHA:AES128-GCM-SHA256:AES1
   28-SHA256:AES128-SHA:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES
   256-SHA:AES256-GCM-SHA384:AES256-SHA256:AES256-SHA:ECDHE-RSA-AES128-SHA256:!aNULL:!eNULL
   :!RC4:!EXPORT:!DES:!3DES:!MD5:!DSS:!PKS;
     ssl_certificate    /home/admin/cai/certs/wildcard.crt;
     ssl_certificate_key /home/admin/cai/certs/wildcard.key;
     ssl_session_cache   shared:SSL:10m;
     ssl_session_timeout 10m;
     location / {
       proxy_pass http://gateway_upstream;
     }
   }
   ```

2. Place the files at specified locations.

   Copy and paste the wildcard.conf file to the /alidata/sites/conf directory of the host for apigateway.ApiGatewayLite#. Copy and paste the wildcard.crt and wildcard.key files to the /alidata/sites/certs directory of the host for apigateway.ApiGatewayLite#.

3. Activate the certificate.

   Run `/home/admin/cai/bin/nginxctl reload` in the apigateway.ApiGatewayLite# SR container.

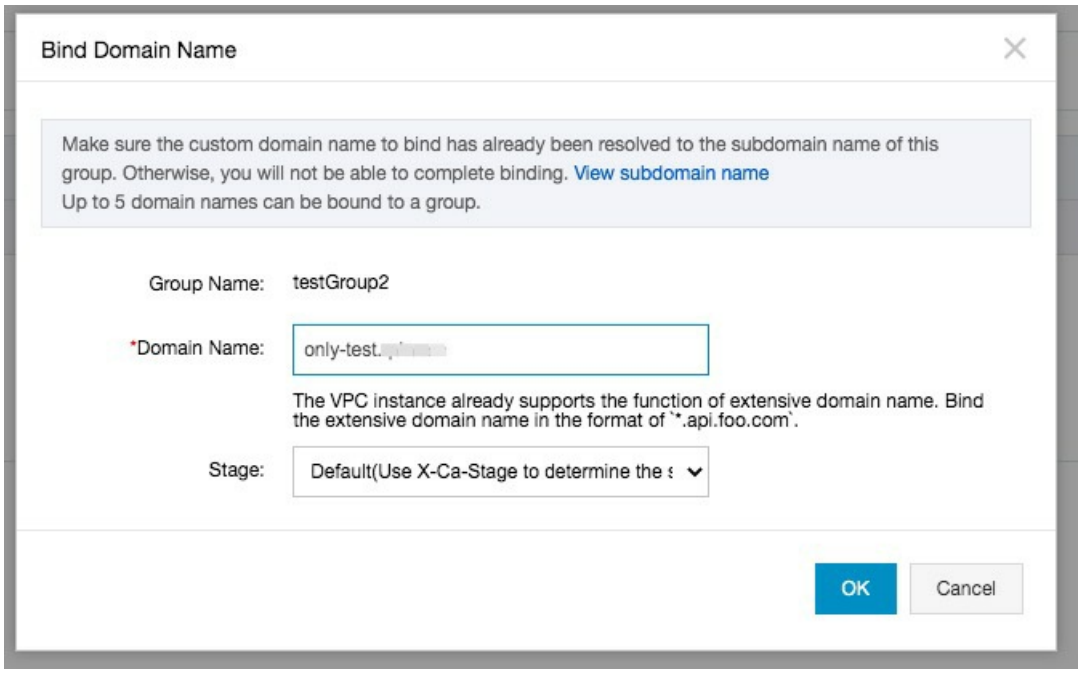
> **Note** You must perform Step 2 and Step 3 on all machines under the apigateway.ApiGatewayLite# SR container.

Use a custom domain name to call APIs over HTTPS

Perform the following steps to use a custom domain name to call APIs over HTTPS:

1. Bind a custom domain name to a specific API group.
   i. Log on to the API Gateway console. In the left-side navigation pane, choose Publish APIs > API Groups. On the Group List page, click the name of the desired group to go to the Group Details page.
   ii. In the Custom Domain Name section, click **Bind Domain**.

   iii. In the Bind Domain Name dialog box, specify Domain Name and click **OK**.



2. Resolve the domain name.

   To access API Gateway by using this domain name, you must resolve the domain name in Apsara Stack to the virtual IP address of API Gateway.

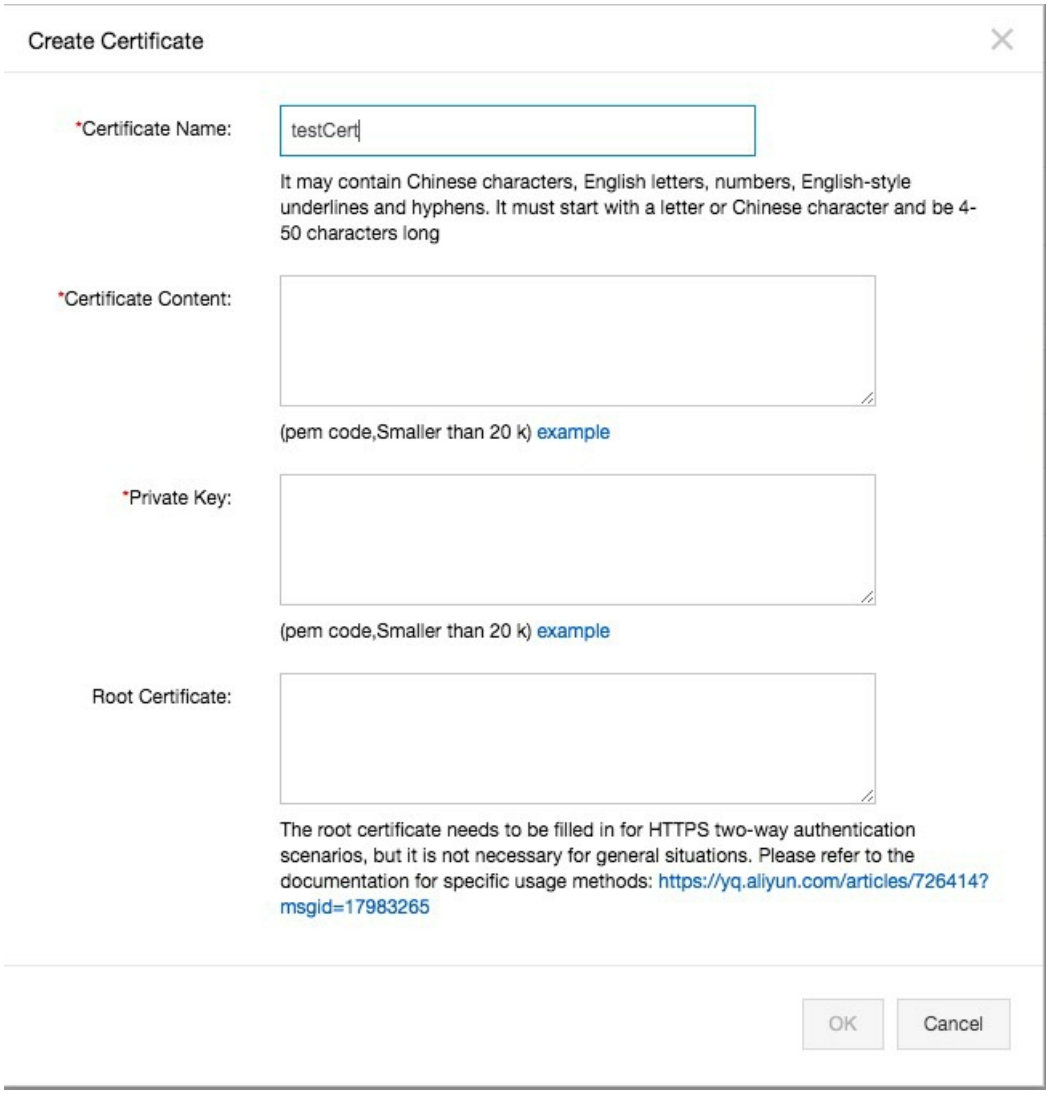
   Run the ping [Second-level domain] command to obtain the virtual IP address of API Gateway.

3. Upload a domain name certificate.

   After a domain name is bound to an API group, you can use the domain name to call all the APIs that belong to this API group over HTTP. If you want to call APIs over HTTPS, you must upload an SSL certificate for the domain name. You must prepare the certificate yourself and upload the certificate to API Gateway.

i. In the **Custom Domain Name** section of the **Group Details** page, find the desired domain name and click **Create Certificate** in the SSL Certificate column.

ii. In the Create Certificate dialog box, specify **Certificate Name**, **Certificate Content**, and **Private Key**. Then, click OK.



Note If the certificate is a self-signed certificate, ignore certificate verification when you call APIs.

1.8. FAQ

1.8.1. How do I obtain error information?

API Gateway returns a response to the client after it receives a request.

You must check the response headers that start with X-Ca. Take note of the following points:

```
// The unique ID of the request. When API Gateway receives a request, it generates a request ID and returns the request ID to the client in the X-Ca-Request-Id header. We recommend that you record the request ID in both the client and your backend service for troubleshooting and tracking.
X-Ca-Request-Id: 7AD052CB-EE8B-4DFD-BBAF-EFB340E0A5AF
// The error message returned by API Gateway. If a request fails, API Gateway returns the error message to the client in the X-Ca-Error-Message header.
X-Ca-Error-Message: Invalid Parameter Required `field1`
// The error code of a system error in API Gateway. If a request is blocked by API Gateway due to an error, API Gateway returns the corresponding error code in the X-Ca-Error-Code header. Instances of the classic network type do not have this header.
X-Ca-Error-Code: I400MP
```

The `X-Ca-Error-Code` and `X-Ca-Error-Message` `headers` help you identify the error cause. The `X-Ca-Request-Id` header helps you query request logs in Log Service. You can also provide the request ID included in the X-Ca-Request-Id header for technical support personnel to check log information and resolve issues.

For more information about `X-Ca-Error-Code`, see Error codes.

1.8.2. Error codes

- If the client receives a response in which the `X-Ca-Error-Code` header is not empty, the header is returned by API Gateway. An error code is six characters in length. For more information, see the following table. `X-Ca-Error-Message` indicates detailed information about an error message.
- If the `X-Ca-Error-Code` header is empty, the HTTP error code is generated by your backend service. API Gateway transparently transmits the error message from your backend service.

Error code	HTTP status code	Error message	Description
I400HD	400	Invalid Header `${HeaderName}` ${Reason}	The error message returned because the HTTP request header is invalid.
I400MH	400	Header `${HeaderName}` is Required	The error message returned because the HTTP request header is missing.

Error code	HTTP status code	Error message	Description
I400BD	400	Invalid Body: ${Reason}	The error message returned because the HTTP request body is invalid.
I400PA	400	Invalid Request Path `${Reason}`	The error message returned because the HTTP request path is invalid.
I405UM	405	Unsupported Method `${Reason}`	The error message returned because the HTTP request method is not supported.
I400RU	400	Invalid Request Uri `${Reason}`	The error message returned because the HTTP request URL is invalid.
I403PT	403	Invalid protocol ${Protocol} unsupported	The error message returned because a protocol that is not supported in API configurations is used. Check the API configurations.
I413RL	413	Request body too Large	The error message returned because the request body is too large.
I413UL	413	Request URL too Large	The error message returned because the request URL is too long.
I400CT	400	Invalid Content-Type: `${Reason}`	The error message returned because the Content-Type setting is invalid.

Error code	HTTP status code	Error message	Description
I404DO	404	Invalid Domain `${DomainName}`	The error message returned because the domain name is unknown.
I410GG	410	Group's instance invalid	The error message returned because an invalid instance is requested. The group may not belong to the current instance.
I400SG	400	Invalid Stage	The error message returned because an unknown environment is requested.
I404NF	404	API not found ${Reason}	The error message returned because the corresponding API is not found based on the Path and Method settings of the request.
X400PM	400	Invalid plugin meta ${PluginName} ${Reason}	The error message returned because the metadata of the plugin is invalid. Submit a ticket to contact customer service.
X500ED	500	Expired api definition	The error message returned because the specified API metadata definition is invalid. Submit a ticket to contact customer service.

Error code	HTTP status code	Error message	Description
X500AM	500	Invalid Api Meta, try deploy again or contact us via ticket	The error message returned because the specified API metadata definition is invalid. Submit a ticket to contact customer service.
X403DG	403	Bad Domain or Group: ${Reason}	The error message returned because the grouped data is invalid. Submit a ticket to contact customer service.
B451DO	451	Unavailable Domain for Legal Reasons	The error message returned because the domain name does not comply with the requirements of relevant laws and regulations.
B451GO	451	Unavailable Group for Legal Reasons	The error message returned because the group does not comply with the requirements of relevant laws and regulations.
B403OD	403	Provider Account Overdue	The error message returned because the API provider has overdue payments.
A400AC	400	Invalid AppCode ${Reason}	The error message returned because the corresponding AppCode is not found when you perform an authorization in AppCode mode.

Error code	HTTP status code	Error message	Description
A400IK	400	Invalid AppKey	The error message returned because the corresponding AppKey setting is not found when you perform an authorization by using a key-secret pair.
A403IS	403	Invalid Signature, Server StringToSign:`${StringToSign}`	The error message returned because the signature is invalid. For more information, see Request signatures.
A403EP	403	App authorization expired	The error message returned because the authorization expired.
A403PR	403	Plugin Authorization Needed	The error message returned because plugin authorization is not performed.
A400MA	400	Need authorization, `X-Ca-Key` or `Authorization: APPCODE ...` is required	The error message returned because authorization is not performed in AppCode mode or by using a key-secret pair.
I400I5	400	Invalid Content-MD5 ${Reason}	The error message returned because Content-MD5 is invalid.
I400NC	400	X-Ca-Nonce is required	The error message returned because the X-Ca-Nonce header is not provided after you select Force Nonce Check (Anti Replay by X-Ca-Nonce).

Error code	HTTP status code	Error message	Description
S403NU	403	Nonce Used	The error message returned because a replay attack is detected. The X-Ca-Nonce header in the request is repeated.
S403TE	403	X-Ca-Timestamp is expired	The error message returned because the timestamp specified by the X-Ca-Timestamp header expired.
I400MP	400	Parameter `${ParameterName}` is required	The error message returned because the required parameter is not specified in the API configuration.
I400IP	400	Invalid parameter `${ParameterName}` ${Reason}	The error message returned because the value of the parameter that is specified in the API configuration is invalid.
I400JR	400	JWT required	The error message returned because no JWT-related parameters are found.
S403JI	403	Claim `jti` is required when `preventJtiReplay:true`	The error message returned because no valid jti claim is included in the request when preventJtiReplay is set to true in a JWT authentication plugin.

Error code	HTTP status code	Error message	Description
S403JU	403	Claim `jti` in JWT is used	The error message returned because the jti claim that is included in the request is used when preventJtiReplay is set to true in a JWT authentication plugin.
I400JD	400	JWT Deserialize Failed: `${Token}`	The error message returned because the JWT that is read from the request failed to be parsed.
A403JT	403	Invalid JWT: ${Reason}	The error message returned because the JWT that is included in the request is invalid.
A403JK	403	No matching JWK, `${kid}` not found	The error message returned because no JWK matches kid configured in the JWT included in the request.
A403JE	403	JWT is expired at `${Date}`	The error message returned because the JWT that is read from the request expired.
I400JP	400	Invalid JWT plugin config: ${JWT}	The error message returned because the JWT authentication plugin is incorrectly configured.

Error code	HTTP status code	Error message	Description
A403OL	403	OAuth2 Login failed: ${Reason}	
A403OU	403	OAuth2 Get User Info failed: ${Reason}	
A401OT	401	Invalid OAuth2 Access Token	
A401OM	401	OAuth2 Access Token is required	
T429ID	429	Throttled by INNER DOMAIN Flow Control, ${Domain} is a test domain, only 1000 requests per day	The error message returned because the number of requests initiated has exceeded the upper limit allowed for a default second-level domain. To increase the quota, use your own domain name.
T429IN	429	Throttled by INSTANCE Flow Control	The error message returned because throttling is performed for the current instance.
T429GR	429	Throttled by GROUP Flow Control	The error message returned because throttling is performed for the current group.
T429PA	429	Throttled by API Flow Control	The error message returned because the default API-level throttling policy defined in the throttling plugin is used.

Error code	HTTP status code	Error message	Description
T429PR	429	Throttled by PLUGIN Flow Control	The error message returned because the special throttling policy defined in the throttling plugin is used.
T429UP	429	Throttled by Usage Plan Flow Control	The error message returned because throttling is performed for the usage plan.
T429SR	429	Throttled by SERVER Flow Control	
T429MR	429	Too Many Requests, throttle by `${Description}`	
A403IP	403	Access denied by IP Control Policy	The error message returned because access is denied by the IP address-based access control plugin.
A403IN	403	Access from internet is disabled ${Reason}	The error message returned because you are not allowed to call APIs or access API groups over the Internet.
A403VN	403	Access from invalid VPC is disabled	The error message returned because access over a VPC is denied.
A403AC	403	Access Control Forbidden by ${RuleName}	The error message returned because access is denied by the access control plugin.

Error code	HTTP status code	Error message	Description
A403CO	403	Cross origin resource forbidden ${Domain}	The error message returned because access is denied by the CORS plugin.
I404CO	404	Cross origin resource not found ${Method} - ${Path}	The error message returned because the API definition is not found based on the Path and Method settings of the request that is pre-checked by the CORS plugin.
I404CH	404	Content not cached, with `Cache-Control:only-if-cached`	
I404NR	404	${Resource} not found	
I404SR	404	Stage route missing: ${Reason}	
B403MO	403	Api Market Subscription overdue	The error message returned because the API provider has overdue payments.
B403MQ	403	Api Market Subscription quota exhausted	The error message returned because the API quota you purchased in Alibaba Cloud Marketplace has been exhausted.
B403ME	403	Api Market Subscription expired	The error message returned because the API subscription relationship expired.

Error code	HTTP status code	Error message	Description
B403MI	403	Api Market Subscription invalid	The error message returned because the API marketplace subscription relationship is invalid.
D504RE	504	Backend domain `${Domain}` resolve failed	The error message returned because the domain name failed to be resolved at the backend.
D504IL	504	Backend domain `${Domain}` resolve to illegal address `${Address}`	The error message returned because the domain name resolution results are invalid at the backend.
D504CO	504	Backend service connect failed `${Reason}`	The error message returned because the backend connection failed. Check the security group configurations, the startup status of the backend server, and firewall configurations.
D504CS	504	Backend http ssl connect failed `${Reason}`	The error message returned because the backend connection over HTTPS failed. Check whether the backend protocol matches the port.
D504TO	504	Backend service request timeout	The error message returned because the backend request timed out.

Error code	HTTP status code	Error message	Description
X504VE	504	Backend service vpc mapped failed	The error message returned because the VPC mapping at the backend is invalid. Submit a ticket to contact customer service.
D503BB	503	Backend circuit breaker busy	The error message returned because the API is protected by its circuit breaker.
D503CB	503	Backend circuit breaker open, ${Reason}	The error message returned because the circuit breaker is open for the API. Check the backend performance of the API.
I508LD	508	Loop Detected	The error message returned because loopback call is detected.
I404DD	404	Device id ${DeviceId} not found	The error message returned because the device ID is not found when you call APIs over WebSocket.
A403FC	403	Function Compute AssumeRole failed ${RequestId}:${Reason}	The error message returned because an authorization error occurs when Function Compute serves as the backend service.
D502FC	502	Function Compute response invalid: ${Reason}	The error message returned because responses from the backend service of the Function Compute type are invalid.

Error code	HTTP status code	Error message	Description
X500ER	500	Service Internal Error	The error message returned because an internal server error occurred. Submit a ticket to contact customer service.
X503BZ	503	Service Busy	The error message returned because the service is busy in API Gateway. Try again later or submit a ticket to contact customer service.
X504TO	504	Service timeout	The error message returned because the service processing timed out in API Gateway.

Some error codes may change with version updates or the addition of new features.