ALIBABA CLOUD

Alibaba Cloud

API Gateway

Security

Document Version: 20220322

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.Enable HTTPS for an API operation

Based on HTTP and the Secure Sockets Layer (SSL) protocol, HTTPS is used to encrypt information and data to secure data transmission. HTTPS is widely used today.

API Gateway supports HTTPS-based encryption of API requests. When you configure an API operation, you can specify that the API operation supports HTTP requests, HTTPS requests, or both.

If you want an API operation to support HTTPS requests, perform the following steps:

Step 1: Make preparations

Prepare the following items:

- An independent domain name.
- An SSL certificate that is applied for the independent domain name.
- A custom certificate that is converted from the SSL certificate. The content and the private key files of the custom certificate must be in the PEM format. For more information, see Certificate format. The Tengine service that is used by API Gateway is based on NGINX and PEM is the only certificate format that is supported by NGINX. Therefore, API Gateway also supports only the PEM certificate format.

An SSL certificate contains two files: XXXXX.key and XXXXX.pem, both of which can be opened in a text editor. The following code snippets show examples of the KEY file and the PEM file of an SSL certificate:

KEY:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA8GjIleJ7rlo86mtbwcDnUfqzTQAm4b3zZEo1aKsfAuwcvCud
....
-----END RSA PRIVATE KEY-----
```

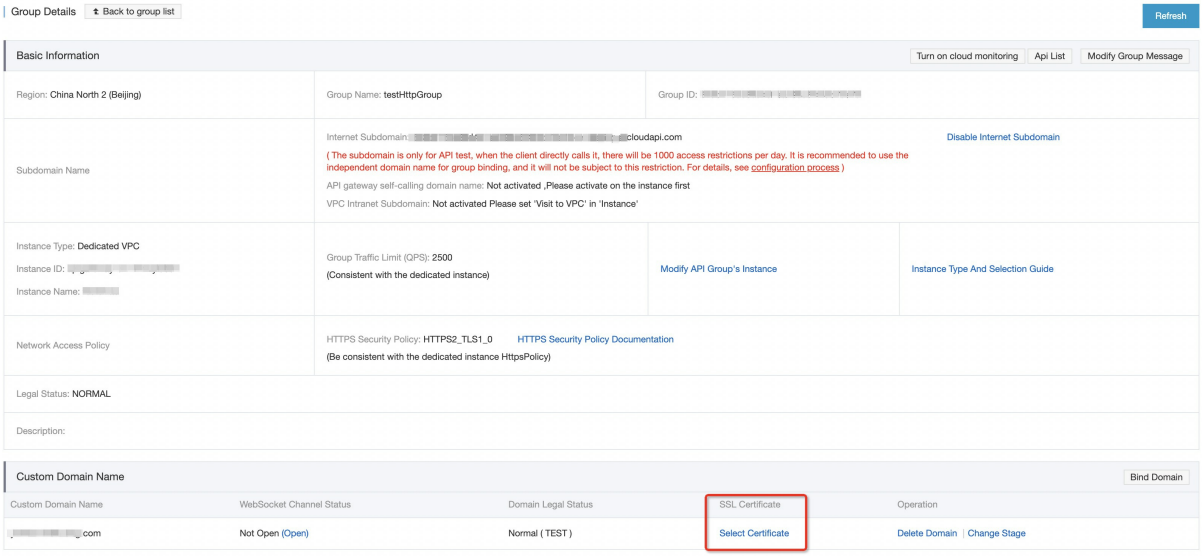
PEM:

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

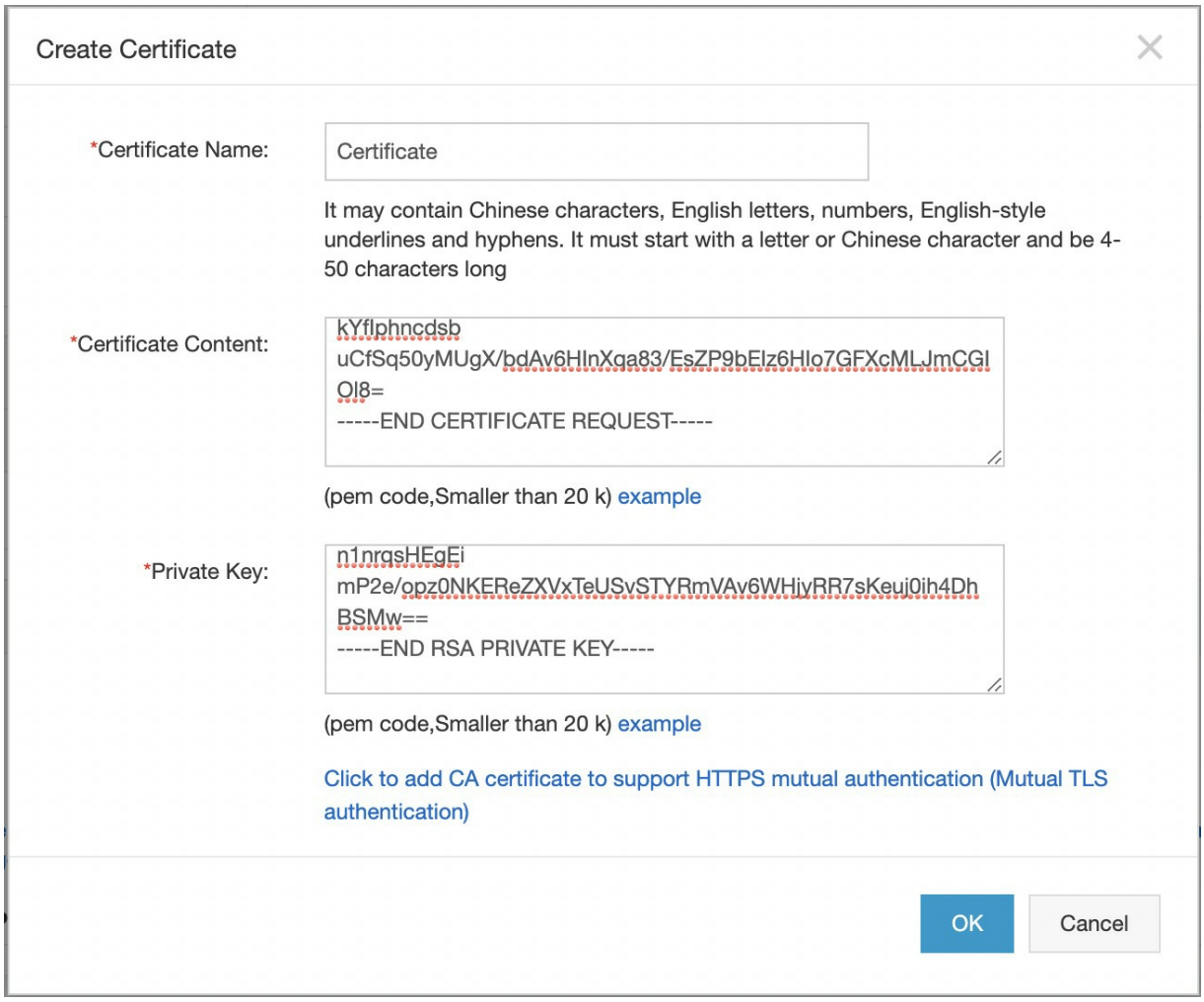
Step 2: Bind the SSL Certificate to an API group

Log on to the API Gateway console. In the left-side navigation pane, choose Publish APIs > API Groups. On the Group List page, find the target API group and click the group name. The Group Details page appears.

In the Custom Domain Name section, bind an independent domain name to the API group.



After you bind the independent domain name, click Select Certificate in the SSL Certificate column. In the Select Certificate dialog box, click Create Certificate. In the Create Certificate dialog box, set relevant parameters, as shown in the following figure.



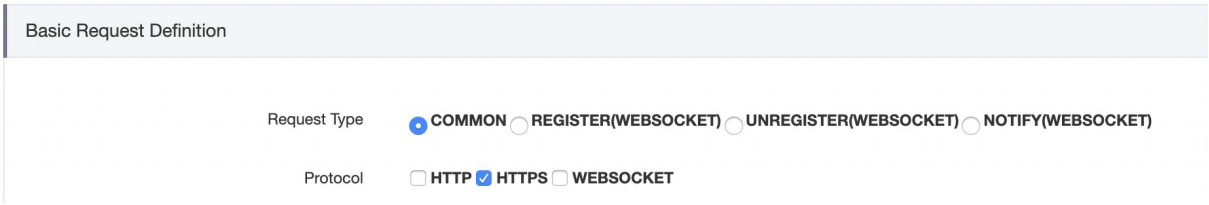
- Certificate Name: the name of the certificate. We recommend that you set an informative name for

easy identification.

- Certificate Content: the complete content of the certificate. Copy the content in the XXXXX.pem file to this field.
- Private Key: the private key of the certificate. Copy the content in the XXXXX.key file to this field. Click OK.

Step 3: Adjust the API configuration

After you bind the SSL certificate to the API group, you can adjust the Protocol parameter that is configured for the API operation. Valid values of the Protocol parameter are HTTP, HTTPS, and WEBSOCKET. You can select one or more protocols for each API operation. We recommend that you set the Protocol parameter to HTTPS for security considerations.



In the left-side navigation pane, choose Publish APIs > APIs. On the API List page, find the target API operation and click its name. On the API Definition page, click Edit in the upper-right corner. In the wizard that appears, go to the Define API Request step.

You can set the **Protocol** parameter to the following values:

- HTTP: supports only HTTP requests.
- HTTPS: supports only HTTPS requests.
- HTTP and HTTPS: support both HTTP and HTTPS requests.

Set the Protocol parameter to HTTPS so that the API operation supports only HTTPS requests.

2.Configure an HTTPS security policy

Configure an HTTPS security policy for an API group

API Gateway allows you to configure HTTPS security policies for an API group, provided that you have bound an independent domain name and a Secure Sockets Layer (SSL) certificate to the API group. API Gateway supports the `HTTPS1_1_TLS1_0`, `HTTPS2_TLS1_0`, and `HTTPS2_TLS1_2` security policies. Note that each region supports different security policies. To view which security policies are supported in the region where an API group resides, log on to the API Gateway console and go to the **Group Details** page of the API group.

Supported HTTPS security policies

HTTPS1_1_TLS1_0

- An HTTP/1.1 protocol.
- Supported Transport Layer Security (TLS) protocol versions: TLS 1.0, TLS 1.1, and TLS 1.2.
- Supported encryption algorithm suite: ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-SHA:AES128-GCM-SHA256:AES128-SHA256:AES128-SHA:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:AES256-GCM-SHA384:AES256-SHA256:AES256-SHA:ECDHE-RSA-AES128-SHA256:!aNULL:!eNULL:!RC4:!EXPORT:!DES:!3DES:!MD5:!DSS:!PKS;

HTTPS2_TLS1_0

- An HTTP/2 protocol. **Note that HTTP/2 converts all header field names to lowercase.**
- Supported TLS protocol versions: TLS 1.0, TLS 1.1, and TLS 1.2.
- Supported encryption algorithm suite: ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-SHA:AES128-GCM-SHA256:AES128-SHA256:AES128-SHA:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:AES256-GCM-SHA384:AES256-SHA256:AES256-SHA:ECDHE-RSA-AES128-SHA256:!aNULL:!eNULL:!RC4:!EXPORT:!DES:!3DES:!MD5:!DSS:!PKS;

HTTPS2_TLS1_2

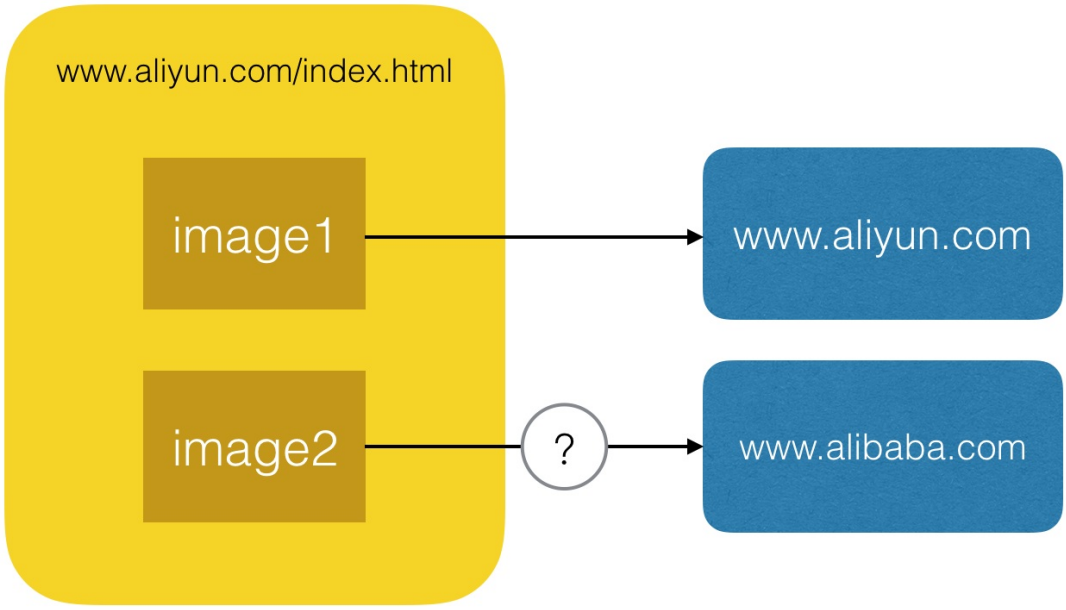
- An HTTP/2 protocol. **Note that HTTP/2 converts all header field names to lowercase.**
- Supported TLS protocol version: TLS 1.2. **Note that after you configure this security policy for an API group, a client can call an API operation in the API group only if the client supports TLS 1.2.**
- Supported encryption algorithm suite: ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA:ECDHE-RSA-AES256-SHA:!NULL:!aNULL:!MD5:!ADH:!RC4:!DH:!DHE:!3DES;

3.Implement CORS in API Gateway

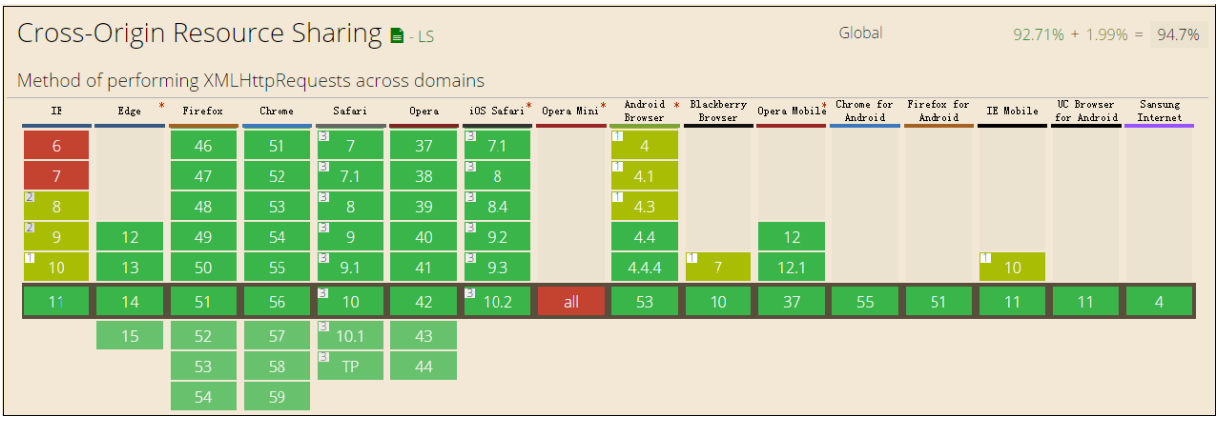
1. Cross-origin resource access: Security risks and browser limits

If a resource on a server requests another resource that is deployed in a different domain or uses a different port, the former resource sends a cross-origin HTTP request. For example, an HTML page from the site http://www.aliyun.com sends a request for an image whose URL is http://www.alibaba.com/image.jpg. Most web pages on the Internet support loading resources, such as CSS, images, and scripts, from different domains.

For security reasons, most browsers forbid sending cross-origin requests from web scripts. Other browsers allow you to send cross-origin requests but responses are blocked. This means that when a web application calls an API operation, only the relevant resources in the same domain can be loaded. To load resources from a different domain, you must configure cross-origin resource sharing (CORS) for the API operation. In this way, the destination server where the requested resource resides will authorize the cross-origin request.



The preceding figure shows a typical scenario of cross-origin resource access. By default, mainstream browsers forbid cross-origin resource access for security reasons. However, these browsers support the CORS mechanism that is recommended by the World Wide Web Consortium (W3C). The CORS mechanism is implemented based on a server and a browser. By using this mechanism, you can enable the browser to allow cross-origin requests.



2. CORS overview

2.1 Two request validation modes

The CORS mechanism supports two request validation modes: simple request validation and preflighted request validation.

If a cross-origin request meets **all of the following three conditions**, the CORS mechanism uses simple request validation to process the cross-origin request.

1. The cross-origin request uses one of the following methods:

- GET
- HEAD
- POST

2. The Content-Type header field in the cross-origin request is set to one of the following values:

- application/x-www-form-urlencoded
- multipart/form-data
- text/plain

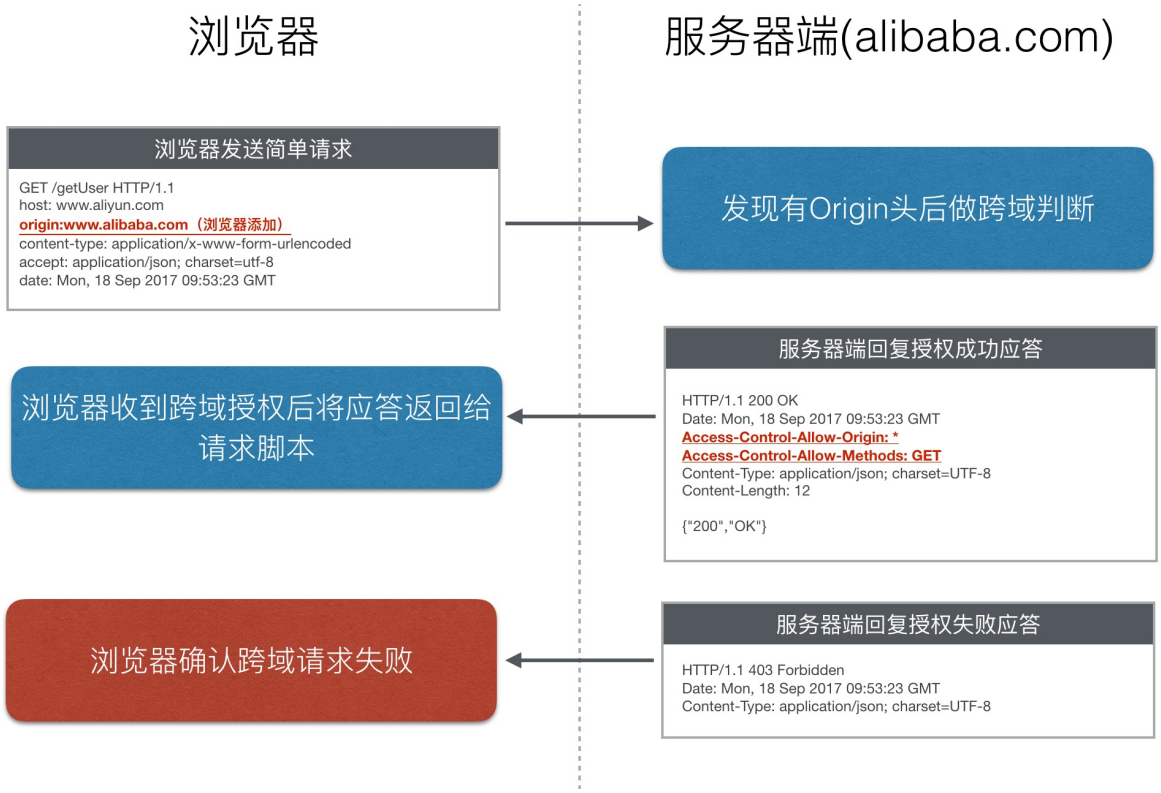
3. The following CORS header fields, including custom header fields in the cross-origin request, are defined in the Fetch standard:

- Accept
- Accept-Language
- Content-Language
- Content-Type (Note that this header field must be set to one of the values that are listed in the second condition.)
- DPR
- Downlink
- Save-Data
- Viewport-Width
- Width

If a cross-origin request does not meet all of the preceding conditions, the CORS mechanism uses preflighted request validation to process the cross-origin request.

2.2 Simple request validation

In the simple request validation mode, a browser sends a cross-origin request. The Origin header field is specified in the request, indicating that the request is a cross-origin request. After the destination server, where the requested resource resides, receives the cross-origin request, the server determines whether to validate the request based on configured CORS rules. If the validation is successful, the server returns a success response to the browser. The success response includes the Access-Control-Allow-Origin and Access-Control-Allow-Methods header fields. If the validation fails, the server returns an error response to the browser.



As shown in the preceding figure, the success response includes the Access-Control-Allow-Origin header field. To sum up, in the simple request validation mode, a cross-origin request from a browser must include the Origin header field and a success response from the destination server must include the Access-Control-Allow-Origin header field. In this example, the value of the Access-Control-Allow-Origin header field in the success response is *, which indicates that the requested resource can be accessed from all domains. If the destination server allows cross-origin requests only from the site http://www.aliyun.com, the value of this header field must be specified as http://www.aliyun.com, as shown in the following code snippet:

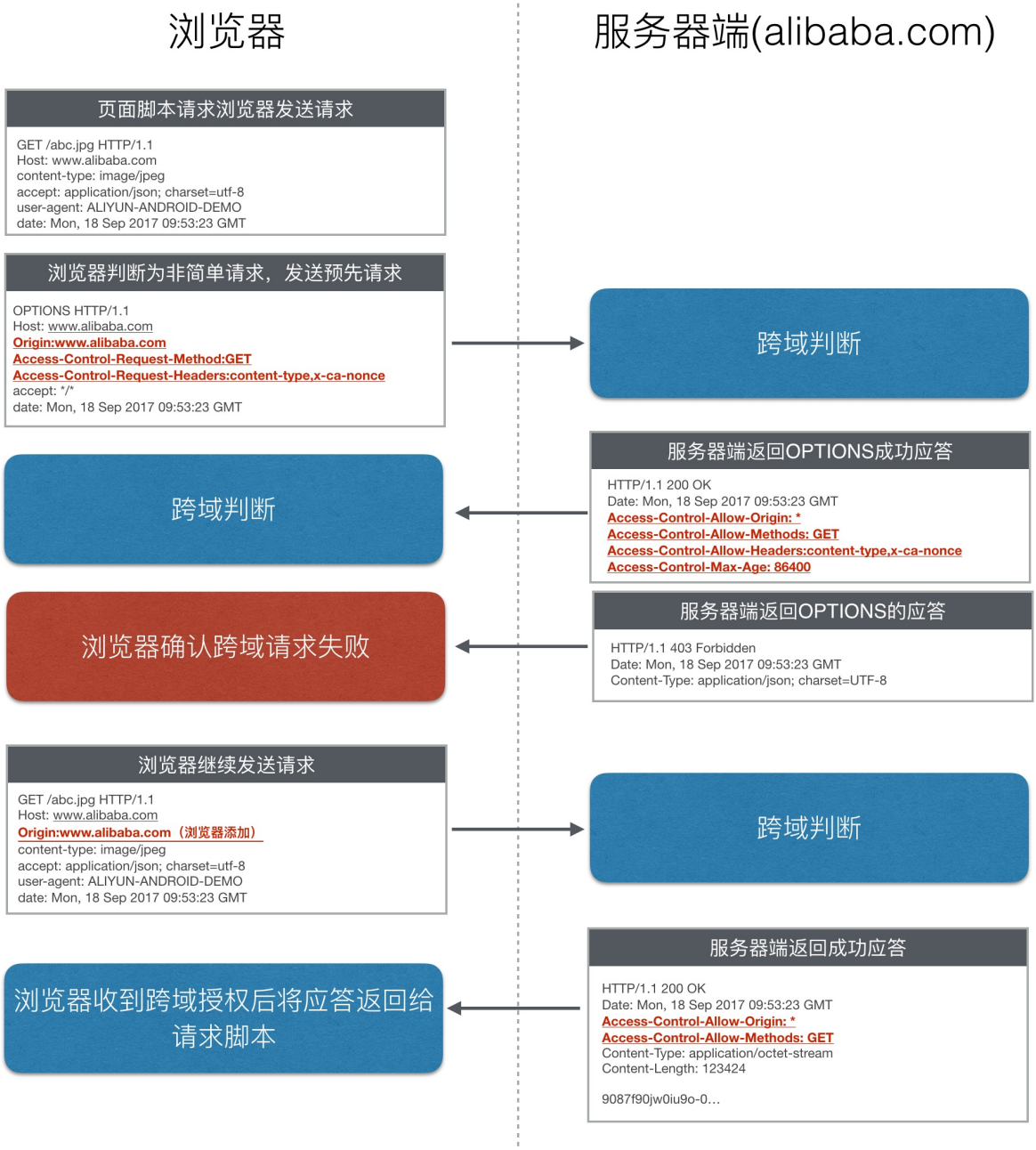
```
Access-Control-Allow-Origin: http://www.aliyun.com
```

In this way, cross-origin requests only from the site http://www.aliyun.com are allowed by the destination server.

2.3 Preflighted request validation

In the preflighted request validation mode, after a browser constructs a cross-origin request, the cross-origin request is not immediately sent to the destination server. Instead, a preflighted request is sent to the destination server. The preflighted request is an HTTP OPTIONS request. This request is used to check whether the destination server, where the requested resource resides, allows cross-origin requests from the current domain name. If the response to the preflighted request indicates that the destination server allows cross-origin requests from the current domain name, the browser then sends the cross-origin request to the server.

The OPTIONS request contains the following header fields: Origin, Access-Control-Request-Method, and Access-Control-Request-Headers. After the destination server receives the OPTIONS request, the server determines whether to validate the preflighted request. If the validation is successful, the server specifies the Access-Control-Allow-Origin, Access-Control-Allow-Method, Access-Control-Allow-Headers, and Access-Control-Max-Age header fields in the success response. After the browser receives the success response to the preflighted request, the browser sends the cross-origin request.



The Access-Control-Request-Method header field in the OPTIONS request informs the destination server that the cross-origin request to be sent uses the GET method. The Access-Control-Request-Headers header field in the OPTIONS request informs the destination server that the cross-origin request to be sent contains two custom header fields: X-Ca-Nonce and Content-Type. Based on these two header fields in the OPTIONS request, the destination server determines whether to allow the cross-origin request.

The Access-Control-Allow-Methods header field in the success response to the OPTIONS request indicates that the destination server allows a cross-origin request from the browser to use the GET method. If multiple methods are allowed, the methods are separated with commas (,).

The Access-Control-Allow-Headers header field in the success response to the OPTIONS request indicates that the destination server allows a cross-origin request to contain the X-Ca-Nonce and Content-Type header fields. The header fields are separated with commas (,).

The Access-Control-Max-Age header field in the success response to the OPTIONS request indicates that the response is valid for 86,400 seconds, namely, 24 hours. Within this validity period, if the browser needs to send the same cross-origin request again, the browser does not need to send another preflighted request. Note that the browser itself specifies a validity period for the cross-origin request. If the value of the Access-Control-Max-Age header field exceeds the validity period that is specified by the browser, the Access-Control-Max-Age header field will not take effect.

3. Implement the CORS mechanism in API Gateway

3.1 Configure the simple request validation mode

By default, all API operations in API Gateway support cross-origin calls. Therefore, by default, API Gateway adds the Access-Control-Allow-Origin header field and specifies its value as * in each API response. The following code snippets show an example of this process:

An API request from a client

```
GET /simple HTTP/1.1
Host: www.alibaba.com
orgin: http://www.aliyun.com
content-type: application/x-www-form-urlencoded; charset=utf-8
accept: application/json; charset=utf-8
date: Mon, 18 Sep 2017 09:53:23 GMT
```

The response that is sent from the backend service of the API operation to API Gateway

```
HTTP/1.1 200 OK
Date: Mon, 18 Sep 2017 09:53:23 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 12
{"200","OK"}
```

The response that is sent from API Gateway to the client

```
HTTP/1.1 200 OK
Date: Mon, 18 Sep 2017 09:53:23 GMT
Access-Control-Allow-Origin: *
X-Ca-Request-Id: 104735BD-8968-458F-9929-DBFA43F324C6
Content-Type: application/json; charset=UTF-8
Content-Length: 12
{"200","OK"}
```

As shown in the preceding code snippets, API Gateway adds specific information to the response that is sent from the backend service, including the following information:

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

API Gateway specifies the Access-Control-Allow-Origin header field as *, which indicates that the API operation can be called from all domains.

If you need to specify the Access-Control-Allow-Origin header field as another value, add the Access-Control-Allow-Origin header field as a response header field when you configure response information for the API operation. The custom value of the Access-Control-Allow-Origin header field will override the default value. The following code snippets show an example in which an API operation can be called only from the site http://www.aliyun.com:

An API request from a client

```
GET /simple HTTP/1.1
Host: www.alibaba.com
orgin: http://www.aliyun.com
content-type: application/x-www-form-urlencoded; charset=utf-8
accept: application/json; charset=utf-8
date: Mon, 18 Sep 2017 09:53:23 GMT
```

The response that is sent from the backend service of the API operation to API Gateway

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://www.aliyun.com
Date: Mon, 18 Sep 2017 09:53:23 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 12
{"200","OK"}
```

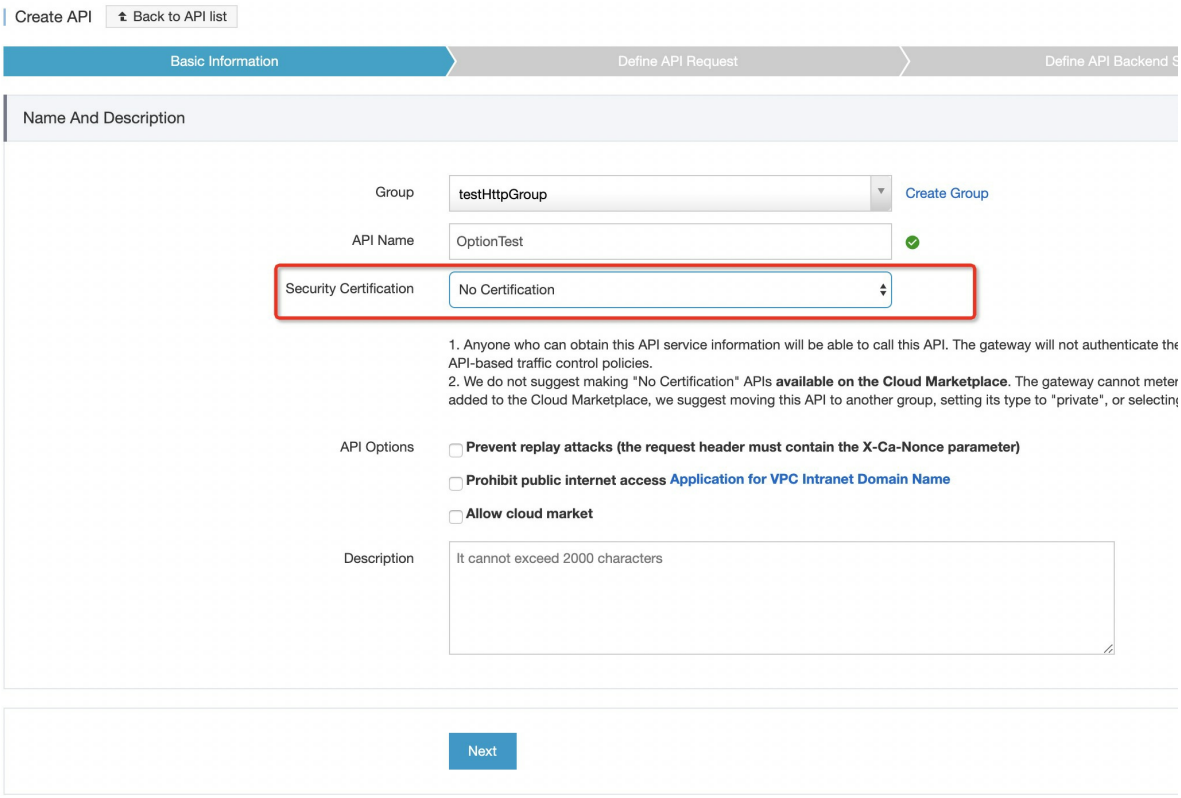
The response that is sent from API Gateway to the client

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://www.aliyun.com
X-Ca-Request-Id: 104735BD-8968-458F-9929-DBFA43F324C6
Date: Mon, 18 Sep 2017 09:53:23 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 12
{"200","OK"}
```

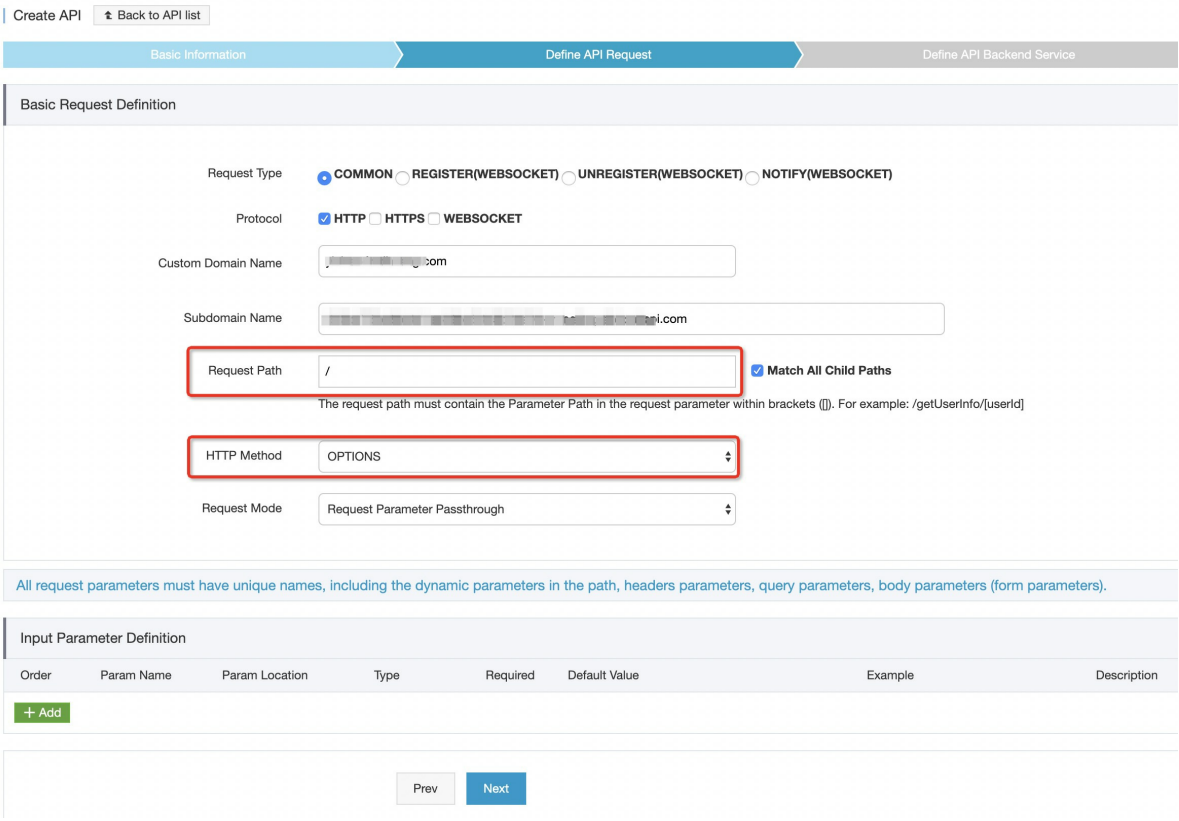
3.2 Configure the preflighted request validation mode

API Gateway allows you to set the HTTP request method of an API operation to OPTIONS. In this case, API Gateway will directly pass each OPTIONS request to the backend service of the API operation. When you create an API operation that allows only OPTIONS requests, take note of the following items:

- When you configure basic information for the API operation, set Security Certification to No Certification.



- When you configure request information for the API operation, enter / in the Request Path field and select the Match All Child Paths check box. After you set HTTP Method to OPTIONS, the Request Mode parameter is automatically set to Request Parameter Passthrough and cannot be modified. You do not need to define request parameters for the API operation.



For each API group, you can create an API operation that allows only OPTIONS requests and use this API operation to configure CORS policies for the API group. You can use curl to test HTTP requests for the API operation that allows only OPTIONS requests. The following code snippet shows an example of using curl to call the API operation:

```
sudo curl -X OPTIONS -H "Access-Control-Request-Method:POST" -H "Access-Control-Request-Hea
ders:X-CUSTOM-HEADER" http://ec12ac094e734544be02c928366b7b26-cn-qingdao.alicloudapi.com/op
tinstest -i
HTTP/1.1 200 OK
Server: Tengine
Date: Sun, 02 Sep 2018 15:32:19 GMT
Connection: keep-alive
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET,POST,PUT,DELETE,HEAD,OPTIONS,PATCH
Access-Control-Allow-Headers: X-CUSTOM-HEADER
Access-Control-Max-Age: 172800
X-Ca-Request-Id: 1016AC86-E345-405C-8049-A6C24078F65F
```

When you configure the API operation that allows only OPTIONS requests, note that API Gateway will add four header fields to each response from the backend service of the API operation: Access-Control-Allow-Origin, Access-Control-Allow-Methods, Access-Control-Allow-Headers, and Access-Control-Max-Age. Therefore, you must add the four header fields as response header fields for the API operation and specify their values as needed, so that the custom values will override the default values.

The following code snippets show an example of a cross-origin API call in the preflighted request validation mode:

An OPTIONS request from a client

```
OPTIONS /simple HTTP/1.1
Host: www.alibaba.com
orgin: http://www.aliyun.com
Access-Control-Request-Method: POST
Access-Control-Request-Headers: X-PINGOTHER, Content-Type
accept: application/json; charset=utf-8
date: Mon, 18 Sep 2017 09:53:23 GMT
```

The response to the OPTIONS request, which is sent from the backend service of the API operation to API Gateway

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://www.aliyun.com
Access-Control-Allow-Methods: GET,POST
Access-Control-Allow-Headers: X-CUSTOM-HEADER
Access-Control-Max-Age: 10000
Date: Mon, 18 Sep 2017 09:53:23 GMT
Content-Type: application/json; charset=UTF-8
```

The response to the OPTIONS request, which is sent from API Gateway to the client

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://www.aliyun.com
Access-Control-Allow-Methods: GET,POST
Access-Control-Allow-Headers: X-CUSTOM-HEADER
Access-Control-Max-Age: 10000
X-Ca-Request-Id: 104735BD-8968-458F-9929-DBFA43F324C6
Date: Mon, 18 Sep 2017 09:53:23 GMT
Content-Type: application/json; charset=UTF-8
```

A cross-origin API request from the client

```
GET /simple HTTP/1.1
Host: www.alibaba.com
orgin: http://www.aliyun.com
content-type: application/x-www-form-urlencoded; charset=utf-8
accept: application/json; charset=utf-8
date: Mon, 18 Sep 2017 09:53:23 GMT
```

The response to the cross-origin request, which is sent from the backend service of the API operation to API Gateway

```
HTTP/1.1 200 OK
Date: Mon, 18 Sep 2017 09:53:23 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 12
{"200","OK"}
```

The response to the cross-origin request, which is sent from API Gateway to the client

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET,POST,PUT,DELETE,HEAD,OPTIONS,PATCH
Access-Control-Allow-Headers: X-Requested-With,X-Sequence,X-Ca-Key,X-Ca-Secret,X-Ca-Version
,X-Ca-Timestamp,X-Ca-Nonce,X-Ca-API-Key,X-Ca-Stage,X-Ca-Client-DeviceId,X-Ca-Client-AppId,X
-Ca-Signature,X-Ca-Signature-Headers,X-Forwarded-For,X-Ca-Date,X-Ca-Request-Mode,Authorizat
ion,Content-Type,Accept,Accept-Ranges,Cache-Control,Range,Content-MD5
Access-Control-Max-Age: 172800
X-Ca-Request-Id: 104735BD-8968-458F-9929-DBFA43F324C6
Date: Mon, 18 Sep 2017 09:53:23 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 12
{"200","OK"}
```

4.JWT-based authentication

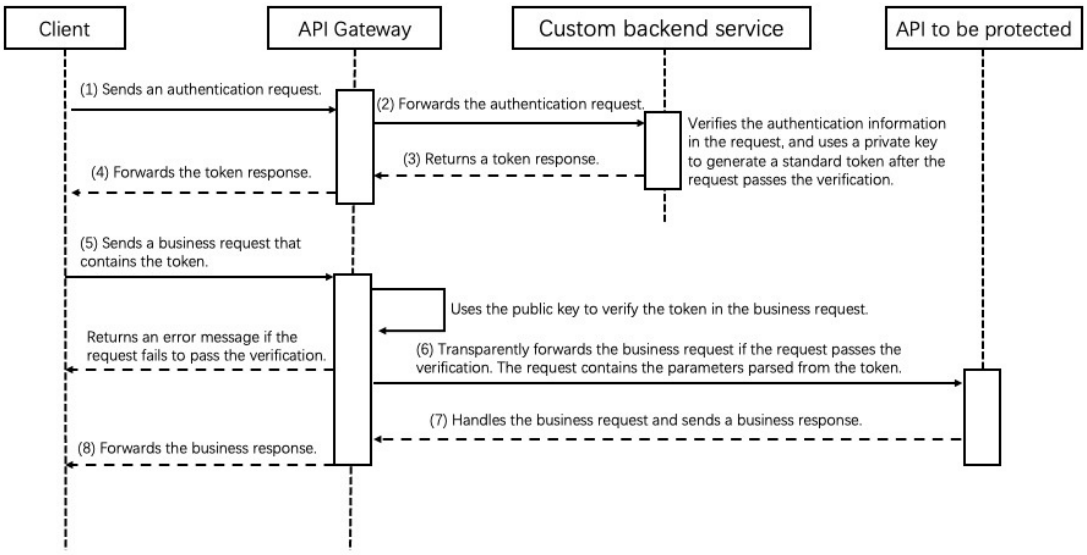
Alibaba Cloud API Gateway provides a mechanism for authorized access to your APIs based on a JSON Web Token (JWT). You can use this mechanism to customize security settings.

1. Token-based authentication

1.1 Overview

API Gateway verifies the identities of requesters who make API calls and determines whether to return requested resources to the requesters. Tokens are a mechanism used for identity authentication. Based on this mechanism, apps do not need to retain user authentication information or session information on the server side. This implements stateless and distributed web app authorization and facilitates app extension.

1.2 Procedure



The preceding figure shows a procedure that is used to implement JWT-based authentication. Detailed procedure description:

1. The client sends an authentication request to API Gateway. In most cases, the username and password of the user are included in the request.
2. API Gateway forwards the authentication request to the backend service.
3. The backend service reads and verifies the authentication information, such as the username and password, in the authentication request. After the request passes the verification, the backend service uses a private key to generate a standard token and returns a token response to API Gateway.
4. API Gateway forwards the token response to the client. The client locally caches the token.
5. The client sends a business request to API Gateway. The token is included in the business request.
6. API Gateway uses a configured public key to verify the token in the business request. If the request passes the verification, API Gateway transparently forwards the business request to the backend service.

7. The backend service handles the business request and sends a business response to API Gateway.
8. API Gateway forwards the business response to the client.

In the procedure, API Gateway allows you to use your own user system to implement token-based API access authentication. The following section describes the structured JWT that is used by API Gateway for authentication.

1.3 JWT

1.3.1 Overview

JWT is an open standard (RFC 7519) that defines a compact and self-contained way to securely transmit information between parties. The information is encoded as a JSON object. A JWT can serve as an independent authentication token, which contains information such as the user ID, user role, and permissions, to help clients obtain resources from the resource server. A JWT can also provide additional claim information required for other business scenarios. This is suitable for logons in distributed sites.

1.3.2 Composition of a JWT

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYW
RtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ
```

As described in the preceding example, a JWT is a string that consists of the following parts:

- Header
- Payload
- Signature

Header

The header consists of the following parts:

- Type of the token, which is JWT
- Encryption algorithm

Example of a complete header in the JSON format:

```
{
  'typ': 'JWT',
  'alg': 'HS256'
}
```

The header is Base64 encoded to form the first part of the JWT. The header you encoded can be symmetrically decoded.

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9
```

Payload

Payload contains valid information specified by a set of claims. Claims:

> *iss: token issuer. This claim is a string. sub: Subject Identifier. The identifier of a user. The value of this claim is unique. This claim can contain a maximum of 255 ASCII characters that are case-sensitive. aud: Audience. Recipients for which the JWT is intended. The value of this claim is a string array that is case-sensitive. exp: Expiration Time. The timestamp at which the token expires. When the timestamp is reached, the token becomes invalid. This claim is an integer representing the number of seconds that have elapsed since the epoch time January 1, 1970, 00:00:00 UTC. iat: the time the token was issued. This claim is an integer representing the number of seconds that have elapsed since the epoch time January 1, 1970, 00:00:00 UTC. jti: the unique identifier of the token. The value of this claim is a cryptographic random value to prevent conflicts. This claim functions in the same way as you add a random entropy component that cannot be obtained by an attacker to the structured JWT. This helps prevent token guessing attacks and replay attacks.*

You can also add custom claims. In the following example, the `name` claim is added:

```
{
  "sub": "1234567890",
  "name": "John Doe"
}
```

The payload is Base64 encoded to form the second part of the JWT.

```
JTdCJTBBJTIwJTIwJTIyc3ViJTIyJTNBJTIwJTIyMTIzNDU2Nzg5MCUyMiUyQyUwQSUyMCUyMCUyMm5hbWUlMjIlM0E
lMjAlMjJKb2huJTIwRG9lJTIyJTBBJTdE
```

Signature

The encryption algorithm specified in the header is used to encrypt the string that is composed of the Base64-encoded header and payload to form the third part of the JWT. The header and payload are concatenated by a period (`.`). `$secret` specifies a private key.

```
// javascript
var encodedString = base64UrlEncode(header) + '.' + base64UrlEncode(payload);
var signature = HMACSHA256(encodedString, '$secret');
```

These three parts are concatenated by periods (`.`) to form a complete string. The JWT is formed.

1.3.3 Authorization scope and validity period

In API Gateway, the issued token can be used to access all APIs that are bound to a JWT authentication plug-in in a specific API group. If fine-grained permission management is required, the backend service must verify the token for authentication. For the validity of a token, API Gateway checks the exp claim in the token in each API request. If the token has expired, API Gateway considers the token invalid and rejects the API request. You must specify a validity period for tokens. The validity period must be less than seven days.

1.3.4 Characteristics of a JWT

1. By default, a JWT is unencrypted. Do not write secret data to the JWT.

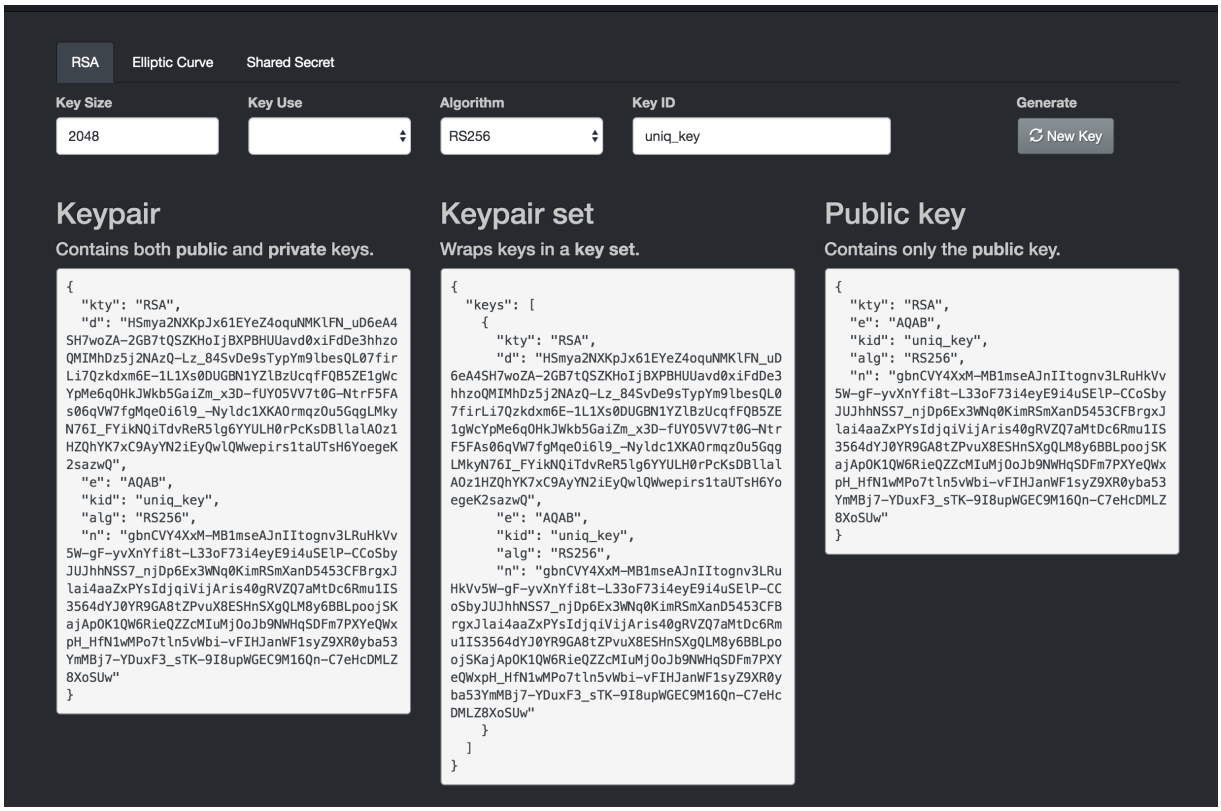
2. A JWT can be used to authenticate identities or exchange information. A JWT can help reduce the number of queries that are performed by the server on a specific database. The most noticeable disadvantage of JWTs is that the server cannot save the session status. Therefore, when a JWT is in use, you cannot revoke it or modify the permissions of the JWT. After a JWT is issued, it is valid until it expires. To invalidate the JWT, you must deploy new logic on the server.
3. A JWT contains authentication information. If the authentication information is disclosed, users can obtain all the permissions of the JWT. To reduce the possibility that a JWT is stolen, set the validity period to a short period for the JWT. Authenticate users who use important permissions.
4. To reduce the possibility that a JWT is stolen, use HTTPS, instead of HTTP, to transmit data.

2. Use a JWT authentication plug-in to protect APIs

2.1 Generate a JWK pair

Method 1: online generation

Visit https://mkjwk.org. Specify a private key and a public key that are used to generate and verify a JWT. The private key is used by an authentication server to issue a JWT. The public key is configured in a JWT authentication plug-in for API Gateway to verify the signature of requests. API Gateway supports the 2048-bit RSA SHA256 encryption algorithm for the key pair.



Method 2: local generation

This topic provides a Java example. Create a Maven project and add the following dependency to the project:

```
<dependency>
     <groupId>org.bitbucket.b_c</groupId>
    <artifactId>jose4j</artifactId>
    <version>0.7.0</version>
</dependency>
```

Use the following code to generate an RSA key pair:

```
RsaJsonWebKey rsaJsonWebKey = RsaJwkGenerator.generateJwk(2048);
rsaJsonWebKey.setKeyId("authServer");
final String publicKeyString = rsaJsonWebKey.toJson(JsonWebKey.OutputControlLevel.PUBLIC_ON
LY);
final String privateKeyString = rsaJsonWebKey.toJson(JsonWebKey.OutputControlLevel.INCLUDE_
PRIVATE);
```

2.2 Use the private key in the JWK pair to issue a token

Use the `Keypair` JSON string that is generated by using method 1 or the `privateKeyString` JSON string that is generated by using method 2 as a private key to issue a token. The token is used to authorize trusted users to access protected APIs. For more information, see the example in the "Sample code for an authentication server to issue a token" section. The form of issuing a token is determined based on specific business requirements. You can deploy the token issuing feature to a production environment and configure a common API. Then, visitors can obtain the token by using a username and password. Alternatively, you can locally generate a token and copy the token for specific users.

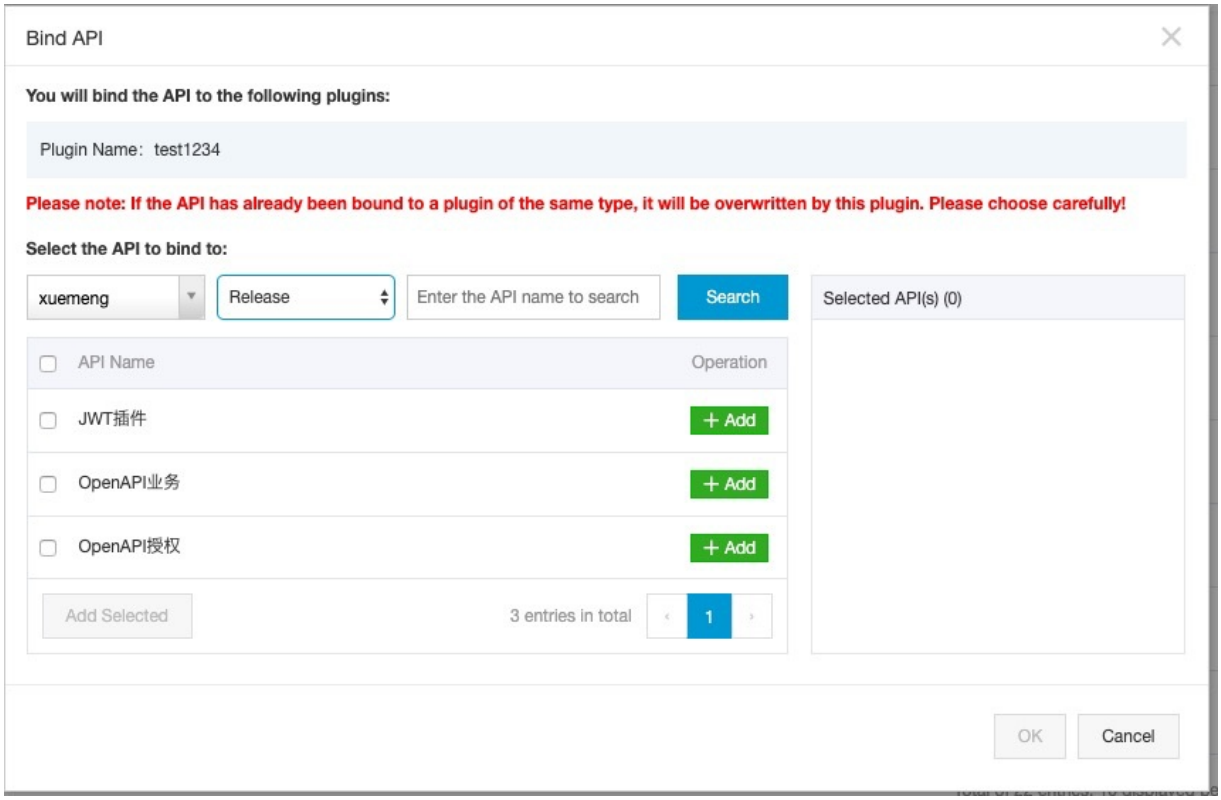
2.3 Configure the public key in the JWK pair for a JWT authentication plug-in

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose Publish APIs > Plugin.
3. In the upper-right corner of the Plugins list page, click `Create Plugin`.
4. On the Create Plugin page, set Plugin Type to `JWT Authorization`. The following example shows the configurations of a JWT authentication plug-in. For more information, see JWT authentication.

```
---
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 a
nd header. This parameter is optional if Request Mode for the bound API is set to Request P
arameter Mapping(Filter Unknown Parameters) or Request Parameter Mapping(Passthrough Unknow
n Parameters). This parameter is required if Request Mode for the bound API is set to Reque
st Parameter Passthrough.
claimParameters:            # The claims to be converted into parameters. API Gateway maps J
WT 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 m
apped.
  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 m
apped.
  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. Defa
ult value: false.
#
# `Public Key` in the `JSON Web Key` pair, which is generated in the "Generate a JWK pair"
section
jwk:
  kty: RSA
  e: AQAB
  use: sig
  alg: RS256
  n: qSVxcknOm0uCq5vGsOmaorPDzHUubBmZZ4UXj-9do7w9X1uKFXAnqfto4TepSNuYU2bA_-tzSLAGBsR-BqvT6w
9SjxakeiyQpVmexxnDw5WZwpWenUAcYrfSPEoNU-0hAQwFYgqZwJQMN8ptxkd0170PFauwACOx4Hfr-9FPGy8NCoIO4
MfLXzJ3mJ7xqgIZp3NIOGXz-GIAbCf13ii7kSStpYqN3L_zzpvXUAos1FJ9IPXRV84tIZpFVh2lmRh0h8ImK-vI42dw
lD_hOIzayL1Xno2R0T-d5AwTSdnep7g-Fwu8-sj4cCRWq3bd61Zs2QOJ8iustH0vSRMYdP5oYQ
```

2.4 Bind APIs to the JWT authentication plug-in

On the Plugins list page, find the JWT authentication plug-in you created and click `Bind API` in the Operation column. In the Bind API dialog box, add the required APIs that are in specific API groups and published to specified environments to the Selected API(s) pane, and click `OK`.



The API debugging feature in the API Gateway console does not support the JWT authentication plug-in. We recommend that you use Postman or run the `curl` command in the command-line interface (CLI) to test the APIs that are bound to the JWT authentication plug-in.

3. Error codes

Status	Code	Message	Description
400	I400JR	JWT required	No JWT-related parameters are found.
403	S403JI	Claim jti is required when preventJtiReplay:true	No valid jti claim is included in the request when preventJtiReplay is set to true in a JWT authentication plug-in.
403	S403JU	Claim jti in JWT is used	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 JWT that is included in the request is invalid.
400	I400JD	JWT Deserialize Failed: ${Token}	The JWT that is read from the request failed to be parsed.
403	A403JK	No matching JWK, kid:${kid} not found	No JWK matches kid, which is configured in the JWT that is included in the request.
403	A403JE	JWT is expired at ${Date}	The JWT that is read from the request expired.
400	I400JP	Invalid JWT plugin config: ${JWT}	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.

4. Sample code for an authentication server to issue a token

```
import java.security.PrivateKey;
import org.jose4j.json.JsonUtil;
import org.jose4j.jwk.RsaJsonWebKey;
import org.jose4j.jwk.RsaJwkGenerator;
import org.jose4j.jws.AlgorithmIdentifiers;
import org.jose4j.jws.JsonWebSignature;
import org.jose4j.jwt.JwtClaims;
import org.jose4j.jwt.NumericDate;
import org.jose4j.lang.JoseException;
public class GenerateJwtDemo {
    public static void main(String[] args) throws JoseException  {
          // Use the value of the keyId parameter that you specified when you configured ba
sic information for the authorization API operation.
        String keyId = "uniq_key";
          // Use the key pair generated in the "Generate a JWK pair" section.
        String privateKeyJson = "{\n"
            + "  \"kty\": \"RSA\",\n"
            + "  \"d\": "
            +
            "\"O9MJSOgcjjiVMNJ4jmBAh0mRHF_TlaVva70Imghtlgwxl8BLfcf1S8ueN1PD7xV6Cnq8YenSKsfi
NOhC6yZ_fjW1syn5raWfj68eR7cjHWjLOvKjwVY33GBPNOvspNhVAFzeqfWneRTBbga53Agb6jjN0SUcZdJgnelzz5J
NdOGaLzhacjH6YPJKpbuzCQYPkWtoZHDqWTzCSb4mJ3n0NRTsWy7Pm8LwG_Fd3pACl7JIY38IanPQDLoighFfo-Lriv
5z3IdlhwbPnx0tk9sBwQBTRdZ8JkqqYkxUiB06phwr7mAnKEpQJ6HvhZBQ1cCnYZ_nIlrX9-I7qomrlE1UoQ\",\n"
```

```
            + "  \"e\": \"AQAB\",\n"
            + "  \"kid\": \"myJwtKey\",\n"
            + "  \"alg\": \"RS256\",\n"
            + "  \"n\": \"vCuB8MgwPZfziMSytEbBoOEwxsG7XI3MaVMoocziP4SjzU4IuWuE_DodbOHQwb_th
Uru57_Efe"
            +
            "--sfATHEa0Odv5ny3QbByqsvjyeHk6ZE4mSAV9BsHYa6GWAgEZtnDceeeDc0y76utXK2XHhC1Pysi2
KG8KAzqDa099Yh7s31AyoueoMnrYTmWfEyDsQL_OAIiwgXakkS5U8QyXmWicCwXntDzkIMh8MjfPskesyli0XQD1AmC
XVV3h2Opm1Amx0ggSOOiINUR5YRD6mKo49_cN-nrJWjtwSouqDdxHYP-4c7epuTcdS6kQHiQERBd1ejdpAxV4c0t0FH
F7MOy9kw\"\n"
            + "}";
        JwtClaims claims = new JwtClaims();
        claims.setGeneratedJwtId();
        claims.setIssuedAtToNow();
        // The validity period is required and must be less than seven days.
        NumericDate date = NumericDate.now();
        date.addSeconds(120*60);
        claims.setExpirationTime(date);
        claims.setNotBeforeMinutesInThePast(1);
        claims.setSubject("YOUR_SUBJECT");
        claims.setAudience("YOUR_AUDIENCE");
        // Add custom parameters. All parameter values must be of the STRING type.
        claims.setClaim("userId", "1213234");
        claims.setClaim("email", "userEmail@youapp.com");
        JsonWebSignature jws = new JsonWebSignature();
        jws.setAlgorithmHeaderValue(AlgorithmIdentifiers.RSA_USING_SHA256);
          // The KeyIdHeaderValue parameter is required.
        jws.setKeyIdHeaderValue(keyId);
        jws.setPayload(claims.toJson());
        PrivateKey privateKey = new RsaJsonWebKey(JsonUtil.parseJson(privateKeyJson)).getPr
ivateKey();
        jws.setKey(privateKey);
        String jwtResult = jws.getCompactSerialization();
        System.out.println("Generate Json Web token , result is " + jwtResult);
    }
}
```

Take note of the following items:

1. The value of the keyId parameter must be unique in API Gateway. You must keep the following values consistent for the keyId parameter:

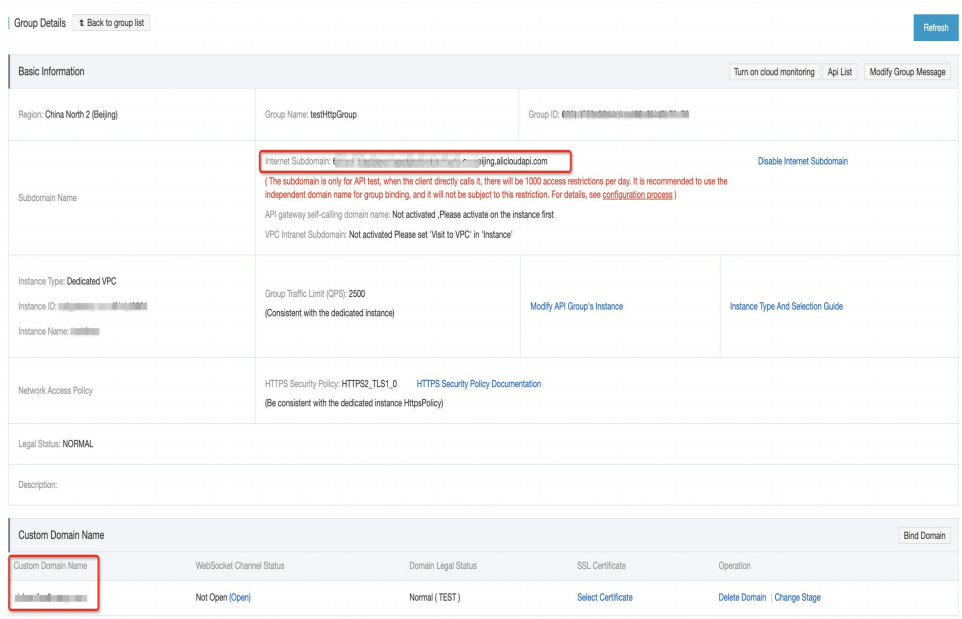
- The value of the keyId parameter that is specified in the "Generate a JWK pair" section.
- The value of the keyId parameter that you specified when you configure basic information for the authorization API operation.
- The value of the keyId parameter that is specified in code, which is the value of the KeyIdHeaderValue parameter in the JsonWebSignature object. The KeyIdHeaderValue parameter is required.

2. You must set privateKeyJson to the `Keypair` JSON string that is generated by using method 1 or the `privateKeyString` JSON string that is generated by using method 2 in the "Generate a JWK pair" section.

3. The validity period is required and must be less than seven days.
4. When you add custom parameters, all parameter values must be of the STRING type.

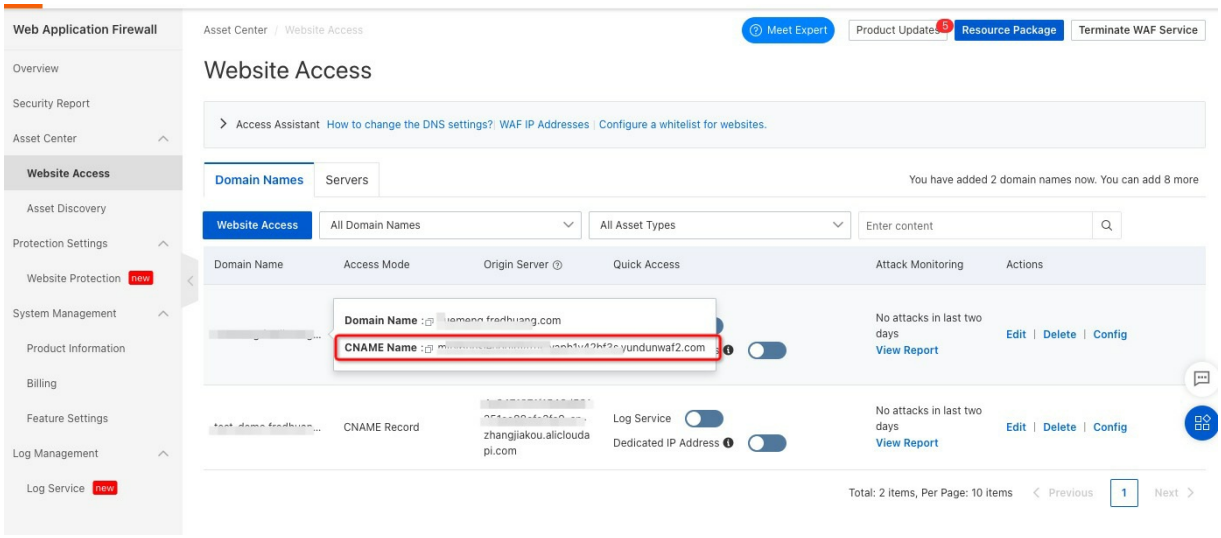
5.Configure WAF

- Purchase a WAF instance
-



Notice

-
-
-


Web Application Firewall
Overview
Security Report
Asset Center
Website Access
Asset Discovery
Protection Settings
Website Protection
new
System Management
Product Information
Billing
Feature Settings
Log Management
Log Service
new
Asset Center / Website Access
Meet Expert
Product Updates
5
Resource Package
Terminate WAF Service
Website Access
Access Assistant How to change the DNS settings? WAF IP Addresses Configure a whitelist for websites.
Domain Names
Servers
You have added 2 domain names now. You can add 8 more
Website Access
All Domain Names
All Asset Types
Enter content
Domain Name
Access Mode
Origin Server
Quick Access
Attack Monitoring
Actions
Domain Name :
CNAME Name :
yundunwaf2.com
No attacks in last two days
View Report
Edit | Delete | Config
CNAME Record
zhangjiakou.alicloudapi.com
Log Service
Dedicated IP Address
No attacks in last two days
View Report
Edit | Delete | Config
Total: 2 items, Per Page: 10 items
Previous
1
Next