API Gateway

User Guide for Consumers

User Guide for Consumers

Call an API

You can use API Gateway to call the API services enabled by other Alibaba Cloud users or third-party service providers. API Gateway provides for you a series of management services and support.

Call example

Based on the SDKs provided by API Gateway, you can write codes to call an API. You can also edit an HTTP request to call an API. The request structure of the API is as follows:

//If the domain name is a13db7999e494a90819cce500130034d.com.

//If the path is /web/cloudapi/mapping/service.

//If the query content is a=name, b=12.

//Then the URL of the request is as follows:

```
http://a13db7999e494a90819cce500130034d.com/web/cloudapi/mapping/service?a=name&amp;b=12
```

//Requesting method.

```
POST HttpMethod: POST
```

//Headers must include signature information and certain parameters.

//For more information about the methods of calculating and passing the encrypted signature, see Portal and Protocol.

```
X-Ca-Version:  1      // API version
X-Ca-Signature-Headers: X-Ca-Version,X-Ca-Key,X-Ca-Stage,X-Ca-Timestamp // Headers involved in signature calculation
X-Ca-Key: 60028305 //AppKey
X-Ca-Stage: test //Stage
X-Ca-Timestamp: 1456905123049 //Time stamp
X-Ca-Signature: UAaH/qteir4G9UK4YR+NWdyq+c1rjl0PvtO/C1Qo68U= // Signature
```

//Standard HTTP header.

```
Host:  a13db7999e494a90819cce500130034d.com        //Service address
Date: Wed 02 Mar 2016 07:52:02 GMT
User-Agent: Apache-HttpClient/4.1.2 (java 1.6)
Content-Type: application/x-www-form-urlencoded; charset=utf-8
```

//Body content.

```
Amount=11&InstanceId=ClientInstanceId&InstanceName=ClientInstanceName
```

An API request is constructed through the preceding content and the inputted parameters of the API. At the public beta stage, you must obtain API documentation and details, such as the service address and path, in the deprecation environment from the API service provider. The AppKey is the key for the created app, which is used for identity verification. The app is your identity to call an API. For more information, see subsequent content.

App

You must create an app as your identity to call an API. Each app has a key pair consisting of an AppKey and AppSecret. These are used as the encrypted signature in your request and is verified by the gateway verifies.

> In API Gateway, create an app as your requester identity. During app creation, the system automatically assigns an AppKey and AppSecret. The AppKey indicates your identity. The AppSecret is the key used to encrypt the signature string and to verify the signature string on the server. When calling an API, you must include the AppKey and AppSecret into the request. API Gateway verifies your identity through symmetric encryption. For more information about the methods of calculating and passing the encrypted signature, see Portal and Protocol.
>
> The AppKey and AppSecret have all of the permissions on the app, and therefore, must be kept secure. If any of the keys are released, you must reset them on the API Gateway console.
>
> You can own multiple apps, to which different APIs are assigned based on your service requirements. Note that the API authorization is specific to an app, but not the Alibaba Cloud user account.

On the API Gateway console, you can manage apps, view details, manage keys, and view authorized apps.

Authorization

Authorization grants an app the permission to call an API. Your app needs authorization for an API before calling it. At the public beta stage, the API service provider establishes the permission relationship between an app and API.

At the public beta stage, the API service provider establishes the authorization. You must provide the API service provider with your AppID or Alibaba Mail account to indicate that an app is given for authorization. After authorization, you can use this app to call the API.

At the public beta stage, you do not have permission to establish or revoke authorization. You can only view the authorized APIs under an app on the console. If you need to revoke the authorization for an API, contact the API service provider.

Encrypted signature

When you call an API, API Gateway uses the AppKey and AppSecret to calculate the encrypted signature for identity verification.

In API Gateway, you must use an app as your identity to call an API. During app creation, the system automatically assigns an AppKey and AppSecret which is used for the server to verify your identity.

Either the HTTP or HTTPS request must include signature information. The AppKey indicates your identity. The AppSecret is used to encrypt and verify the signature string on the server. For more information about the methods of calculating and passing the encrypted signature, see Portal and Protocol.

Limits

Limit items	Can Be Increased	
There can be up to 1000 apps under each account, and each app name must be unique.	Yes	Support Center
The request packet of per API call does not exceed 2MB	No	-
At the public beta stage, you do not have permission to authorize an app or revoke authorization. Only the API service provider has such permissions.	No	-
Your request must include the signature information. For more information, see Portal and Protocol.	No	-

Request signature description

Domain name

- Each API belongs to an API group, and each API group has a unique domain name. These independent domain names are bound by the service provider. API Gateway uses a domain name to locate an API group.
- The domain name is in the format of **www.[Independent domain name].com/[Path]?[HTTPMethod]**. At the public beta stage, the API user needs to obtain this domain name offline from the API service provider.
- Alibaba Cloud API Gateway uses the domain name to locate a unique API group, and then locate the unique API through Path+HTTPMethod.
- You must obtain API documentation in the deprecation environment from the API service provider. This documentation must include necessary parameter information, such as the

domain name and path.

System headers

- [Required] X-Ca-Key: AppKey
- [Required] X-Ca-Signature: Signature string
- [Optional] X-Ca-Timestamp: The time stamp in milliseconds passed by the API caller, that is, the milliseconds of the time from January 1, 1970 until now. By default, it is valid within 15 minutes.
- [Optional] X-Ca-Nonce: The UUID generated by the API caller. This header is used with the time stamp to prevent replay.
- [Optional] Content-MD5: When the request body is not a Form, calculate the MD5 value of the body and send that value to the cloud gateway for checking.
- [Optional] X-Ca-Stage: The stage where the requested API belongs. Only test and release are supported, and the default value is release.

Signature verification

For more information about the demo (Java) of signature calculation, see here.

The signature calculation procedure is as follows:

Organize the strings involved in signature calculation

```
String stringToSign=
HTTPMethod + "\n" +
Accept + "\n" +
Content-MD5 + "\n"
Content-Type + "\n" +
Date + "\n" +
Headers +
Url
```

Each letter of the HTTPMethod value must be capitalized.

If **Accept**, **Content-MD5**, **Content-Type**, and **Date** are empty, add a linefeed (\n). If **Headers** is empty, a linefeed (\n) is not required. The specified **Headers** includes a linefeed (\n). For more information, see the headers organization method described as follows.

Content-MD5

Content-MD5 indicates the MD5 value of the body. MD5 is only calculated when the body is not a Form. The calculation method is as follows:

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

The **bodyStream** indicates the byte array.

Headers

Headers indicates the keys and values of the headers involved in signature calculation. Note that X-Ca-Signature and X-Ca-Signature-Headers are excluded in Headers signature calculation.

Headers organization method:

Rank the keys of all Headers involved in signature calculation in lexicographic order and then splice them in the following method:

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

URL

URL indicates the Form parameter in the Path+Query+Body. The organization method is as follows:

Rank the keys of Query+Form in lexicographic order and then splice them in the following method. If Query or Form is empty, the URL is equal to Path, and a question mark (?) is not required to be added.

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

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

Calculate the signature

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

The **secret** indicates the key corresponding to an app.

Pass the signature

Put the calculated signature in the Header of the Request. The key is X-Ca-Signature.

Separate the keys of all Headers involved in signature calculation by commas and put them in the Header of the Request regardless of the order. The key is X-Ca-Signature-Headers.

For more information about the demo of signature calculation, click here.

Use Postman to sign and debug incoming requests to API Gateway

Use Postman to sign and debug incoming requests to API Gateway

1. Preface

Postman is a powerful HTTP client for testing Web services. Postman is available as a native application for Windows, Mac, and Linux systems. You must sign HTTP requests to an API before you can call the API. Simple HTTP client tools such as Curl cannot be used to sign the requests. However, you can use the Pre-request Script feature of Postman to sign and debug API requests and implement API calls.

2. Signature algorithms for API Gateway

For more information about the request signing mechanism for API Gateway, see the Request signing instructions. Here is a brief introduction.

You must sign incoming requests to API Gateway by using AppKeys and AppSecrets, which can be obtained from the API Gateway console. You must also ensure that the requested APIs have been published and specific applications are authorized to call the APIs.

The procedure for signing a common request to API Gateway is as follows:

2.1. Add the following headers for signature and security authentication:

```
-Date: the date header.
-X-Ca-Key: {AppKey}
-X-Ca-Nonce: the UUID generated by the API caller to prevent replay attacks.
```

-Content-MD5: the MD5 value of the request body. This header is used to verify whether the request body is tampered with when it is not a Form.

2.2. Organize the header elements to be included in StringToSign

```
{HTTPMethod} + "\n" +
{Accept} + "\n" +
{Content-MD5} + "\n"
{Content-Type} + "\n" +
{Date} + "\n" +
{SignatureHeaders} +
{UrlToSign}
```

- Even if Accept, Content-MD5, Content-Type, and Date are not specified, line breaks "\n" are required.
- Content-MD5 is calculated only when the message body is not a Form. The calculation formula is as follows: base64Encode(md5(body.getBytes("UTF-8"))
- SignatureHeaders: Headers to be signed are added in ascending order in the form of {HeaderName}:{HeaderValue} + "\n". Recommended signature headers are X-Ca-Key and X-Ca-Nonce. You can choose other headers to add to the signature.
- UrlToSign: All Form and QueryString fields are placed together and sorted by Name. If Content-Type is not application/x-www-form-urlencoded, all Form fields are treated as a whole. Sorted key-value pairs are appended to Path to obtain UrlToSign. Assume that the query string is /Demo? C = 1 & a = 2 and the form data is B = 3, then UrlToSign=/Demo? a=2&b=3&c=1.

2.3. Calculate the signature and add the headers related to the signature

We recommend that you use the HMAC-SHA256 algorithm to calculate the signature based on the AppSecret. The calculation formula is as follows:signature = base64(hmacSHA256(stringToSign.getBytes("UTF-8), appSecret)). You must add the following headers after the calculation:

- X-Ca-Siguature:{signature}
- X-Ca-SignatureMethod:HmacSHA256
- X-Ca-SignatureHeaders:X-Ca-Key,X-Ca-Nonce

2.4. Troubleshoot signature errors

- If the signature verification fails, API Gateway puts the server-side StringToSign into the HTTP response header and sends the response back to the client. The key is X-Ca-Error-Message. To locate the problem, you must compare the locally calculated StringToSign with the StringToSign returned by the server. Note that the StringToSign returned by the server

replaces the carriage return with #.
- If the signature string is the same on both the client and server, check whether the correct key is used for signature calculation.

3. Use Pre-request Script to implement signature algorithms

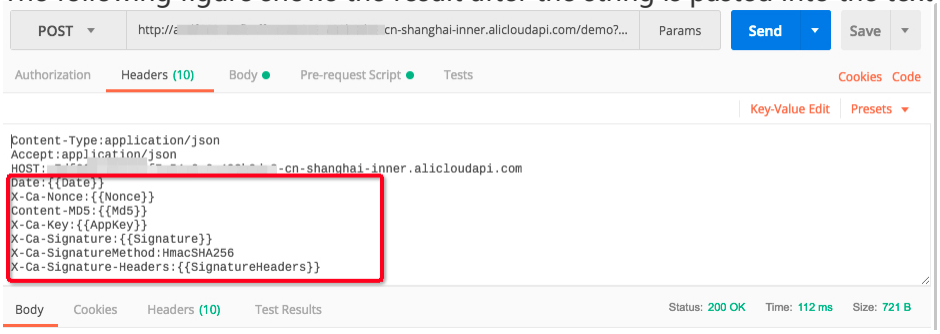
Based on the description in the previous section, the key to debugging API requests is to implement request signature. Postman allows you to use JavaScript code to manipulate the data sent with the request. For more information, see the Pre-request Script development documentation. You can use Pre-request Script to sign API requests.

3.1. Use global variables to preset the signature headers to be added

You cannot modify requests created in Postman scripts. You can add signature headers only by defining global variables for the signature headers and assigning values to the variables. All headers to be signed are preset in your Postman request header. You can switch to the Bulk Edit mode and add these headers, as shown in the following figure. ` ![00_02_28__08_25_2018.jpg](http://ata2-img.cn-hangzhou.img-pub.aliyun-inc.com/d727c94c4c8e06ab51617de54a6cab63.jpg) After switching to theBulk Editmode, you can copy and paste the following string into the text box. The variables enclosed in{{}}` are the global variables in Postman. These variables will be replaced with specific values in scripts. Form content can be left without Content-MD5 headers.

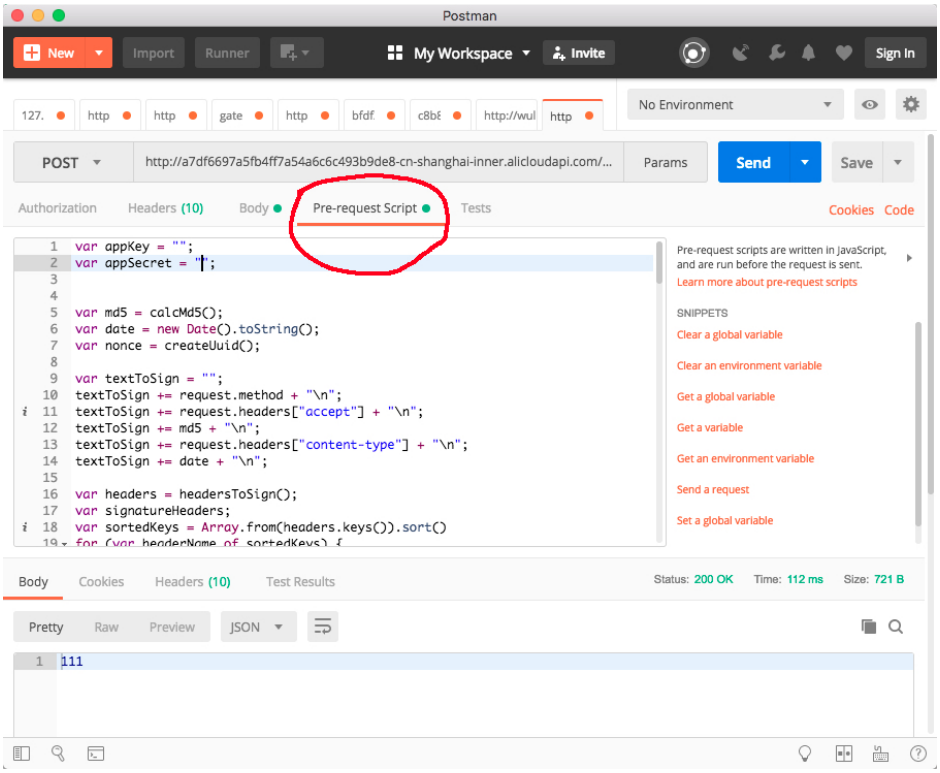
```
Date:{{Date}}
Content-MD5:{{Md5}}
X-Ca-Nonce:{{Nonce}}
X-Ca-Key:{{AppKey}}
X-Ca-Signature:{{Signature}}
X-Ca-SignatureMethod:HmacSHA256
X-Ca-Signature-Headers:{{SignatureHeaders}}
```

The following figure shows the result after the string is pasted into the text box.



3.2. Use Pre-request Script to sign requests

Click the position circled in red to enter pre-request scripts. Copy and paste the following code into the text box:




```
var appKey = "<YOUR APP KEY>";
var appSecret = "<YOUR APP SECRET>";

var md5 = calcMd5();
var date = new Date().toString();
var nonce = createUuid();

var textToSign = "";
textToSign += request.method + "\n";
textToSign += request.headers["accept"] + "\n";
textToSign += md5 + "\n";
textToSign += request.headers["content-type"] + "\n";
textToSign += date + "\n";

var headers = headersToSign();
var signatureHeaders;
var sortedKeys = Array.from(headers.keys()).sort()
for (var headerName of sortedKeys) {
textToSign += headerName + ":" + headers.get(headerName) + "\n";
signatureHeaders = signatureHeaders ? signatureHeaders + "," + headerName : headerName;
}
textToSign += urlToSign();
console.log("textToSign\n" + textToSign.replace(/\n/g, "#"));
var hash = CryptoJS.HmacSHA256(textToSign, appSecret)
console.log("hash:" + hash)
var signature = hash.toString(CryptoJS.enc.Base64)
console.log("signature:" + signature)
```

```
pm.globals.set('AppKey', appKey);
pm.globals.set('Md5', md5);
pm.globals.set("Date", date);
pm.globals.set("Signature", signature);
pm.globals.set("SignatureHeaders", signatureHeaders);
pm.globals.set("Nonce", nonce);

function headersToSign() {
var headers = new Map();
for (var name in request.headers) {
name = name.toLowerCase();
if (! name.startsWith('x-ca-')) {
continue;
}
if (name === "x-ca-signature" || name === "x-ca-signature-headers" || name == "x-ca-key" || name === 'x-ca-
nonce') {
continue;
}
var value = request.headers[name];
headers.set(name, value);
}
headers.set('x-ca-key', appKey);
headers.set('x-ca-nonce', nonce);
return headers;
}

function urlToSign() {
var params = new Map();
var contentType = request.headers["content-type"];
if (contentType && contentType.startsWith('application/x-www-form-urlencoded')) {
const formParams = request.data.split("&");
formParams.forEach((p) => {
const ss = p.split('=');
params.set(ss[0], ss[1]);
})
}

const ss = request.url.split('?') ;
if (ss.length > 1 && ss[1]) {
const queryParams = ss[1].split('&');
queryParams.forEach((p) => {
const ss = p.split('=');
params.set(ss[0], ss[1]);
})
}

var sortedKeys = Array.from(params.keys())
sortedKeys.sort();

var l1 = ss[0].lastIndexOf('/');
var url = ss[0].substring(l1);
var first = true;
var qs
for (var k of sortedKeys) {
var s = k + "=" + params.get(k);
qs = qs ? qs + "&" + s : s;
```

```
console.log("key=" + k + " value=" + params.get(k));
}
return qs ? url + "?" + qs : url;
}

function calcMd5() {
var contentType = request.headers["content-type"];
if (request.data && ! contentType.startsWith('application/x-www-form-urlencoded')) {
var data = request.data;
var md5 = CryptoJS.MD5(data);
var md5String = md5.toString(CryptoJS.enc.Base64);
console.log("data:" + data + "\nmd5:" + md5String);
return md5String;
} else {
return "";
}
}

function createUuid() {
return 'xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx'.replace(/[xy]/g, function(c) {
var r = Math.random()*16|0, v = c == 'x' ? r : (r&0x3|0x8);
return v.toString(16);
});
}
```

Now you can implement debugging through API Gateway.