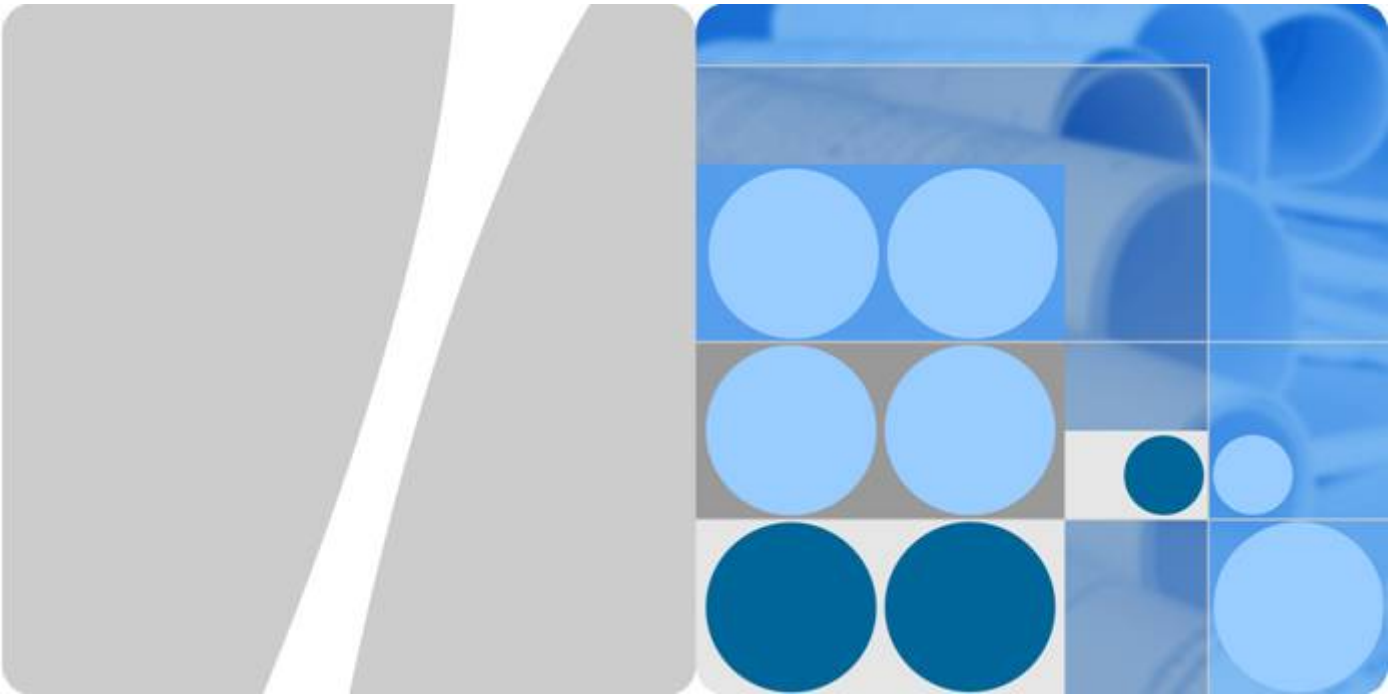


NAT Gateway

API Reference

Issue 03

Date 2018-03-30

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: http://www.huawei.com

Email: support@huawei.com

Contents

1 API Calling

Application programming interface (API) requests sent by third-party applications to public cloud services must be authenticated using signatures.

This chapter describes the signature usage procedure, provides sample code to illustrate how to use the default signer to sign requests and how to use the HTTP client to send requests.

1.1 Service Usage

Public cloud services provide RESTful APIs.

Representational State Transfer (REST) allocates Uniform Resource Identifiers (URIs) to dispersed resources so that resources can be located. Applications on clients use Uniform Resource Locators (URLs) to obtain resources.

The URL is in the following format: https://*Endpoint*/*uri*

Table 1-1 describes the parameters in a URL.

Table 1-1 Parameter description

Parameter	Description
Endpoint	Specifies the URL that is the entry point for a web service. Obtain the value from **Regions and Endpoints**.
URI	Specifies the API access path for performing a specified operation. Obtain the value from the URI of the API, for example, **v3/auth/tokens**.

1.2 Request Methods

The HTTP protocol defines request methods, such as GET, PUT, POST, DELETE, and PATCH, to indicate the desired action to be performed on the identified resource. The following table describes the HTTP methods supported by the RESTful APIs.

Table 1-2 HTTPS methods

Method	Description
GET	The GET method requests a representation of the specified resource.
PUT	The PUT method requests that the enclosed entity be stored under the supplied URI.
POST	The POST method requests that the server accept the entity enclosed in the request as a new subordinate of the web resource identified by the URI.
DELETE	The DELETE method deletes the specified resource, for example, an object.
PATCH	The PATCH method applies partial modifications to a resource. If the resource does not exist, the PATCH method creates a resource.

1.3 Request Authentication Methods

You can use either of the following two authentication methods to call APIs:

- Token authentication: Requests are authenticated using Tokens.
- AK/SK authentication: Requests are encrypted using the access key (AK) and secret key (SK) to provide higher security.

1.4 Token Authentication

Scenarios

If you use a token for authentication, you must obtain the user's token and add **X-Auth-Token** to the request message header of the service API when making an API call.

This section describes how to make an API call for token authentication.

Make an API Call

1. Send POST https://*Endpoint of IAM*/v3/auth/tokens to obtain the endpoint of IAM and the region name in the message body.

   See **Regions and Endpoints**.

   If the service region name is **ALL**, replace *Endpoint of IAM* with the endpoint of **CN North-Beijing1**.

   An example request message is as follows:

   NOTE

   Replace the items in italic in the following example with actual ones. For details, see the *Identity and Access Management API Reference*.

```
{
  "auth": {
    "identity": {
      "methods": [
        "password"
```

```
      ],
      "password": {
        "user": {
          "name": "username",
          "password": "password",
          "domain": {
            "name": "domainname"
          }
        }
      }
    },
    "scope": {
      "project": {
        "id": "0215ef11e49d4743be23dd97a1561e91" //This ID is used as an
example.
      }
    }
  }
}
```

2. Obtain the token. For details, see section "Obtaining the User Token" in the *Identity and Access Management API Reference*.
3. Make a call to a service API, add **X-Auth-Token** to the message header, and set the value of **X-Auth-Token** to the token obtained in step 2.

1.5 AK/SK Authentication

When you use API Gateway to send requests to underlying services, the requests are signed using the AK and SK.

NOTE

AK: indicates the ID of the access key. AK is used together with SK to obtain an encrypted signature for a request.

SK: indicates the secret access key together used with the access key ID to sign requests. AK and SK can be used together to identify a request sender to prevent the request from being modified.

When the AK/SK of a user under a domain is used, add **X-Domain-Id** to the header and set it to the user's domain ID.

1.5.1 AK and SK Generation

1. Log in to the management console.
2. Hover the mouse over the username and select **Basic Information** from the drop-down list.
3. On the **Account Info** page, click **Manage** after **Security Credentials**.
4. On the **My Credential** page, click **Access Keys**.
5. Click **Add Access Key**.
6. Enter the current login password.
7. Enter the authentication code received in the email or mobile phone.

   NOTE

   For users created in Identity and Access Management (IAM), if no email address or mobile phone is filled during the user creation, you only need to authenticate the login password.

8. Click **OK** to download the access key.

NOTE

To prevent the access key from being leaked, keep it secure.

1.5.2 Request Signing Procedure

Preparations

1. Download the API Gateway signature tool.

   Download path: **http://esdk.huawei.com/ilink/esdk/download/HW_456706**
2. Extract the package.
3. Create a Java project, and reference the extracted JAR to the dependency path.

Sign a Request

1. Create a request **com.cloud.sdk.DefaultRequest (JAVA)** used for signing.
2. Set the target API URL, HTTPS method, and content of request **com.cloud.sdk.DefaultRequest (JAVA)**.
3. Sign request **com.cloud.sdk.DefaultRequest (JAVA)**.
   a. Call **SignerFactory.getSigner(String serviceName, String regionName)** to obtain a signing tool.
   b. Call **Signer.sign(Request<?> request, Credentials credentials)** to sign the request created in step 1.

      The following code shows the details:

      ```
      //Select an algorithm for request signing.
      Signer signer = SignerFactory.getSigner(serviceName, region);
      //Sign the request. The request will change after the signing.
      signer.sign(request, new BasicCredentials(this.ak, this.sk));
      ```
4. Convert the request signed in the previous step to a new request that can be used to make an API call and copy the header of the signed request to the new request.

   For example, if Apache HttpClient is used, convert DefaultRequest to HttpRequestBase and copy the header of the signed DefaultRequest to HttpRequestBase.

   For details, see descriptions of AccessServiceImpl.java in section **1.5.3 Sample Code**.

1.5.3 Sample Code

The following three types of code show how to sign a request and how to use an HTTP client to send an HTTPS request:

AccessService: indicates the abstract class that converts the GET, POST, PUT, and DELETE methods in to the access method.

Demo: indicates the execution entry used to simulate GET, POST, PUT, and DELETE request sending.

AccessServiceImpl: indicates the implementation of the **access** method. Code required for API gateway communication is in the access method.

For details about **region** and **serviceName** in the following code, see **Regions and Endpoints**.

AccessService.java:

```
package com.cloud.apigateway.sdk.demo;

import java.io.InputStream;
import java.net.URL;
import java.util.Map;

import org.apache.http.HttpResponse;

import com.cloud.sdk.http.HttpMethodName;

public abstract class AccessService {

    protected String serviceName = null;

    protected String region = null;

    protected String ak = null;

    protected String sk = null;

    public AccessService(String serviceName, String region, String ak, String sk)
{
        this.region = region;
        this.serviceName = serviceName;
        this.ak = ak;
        this.sk = sk;
    }

    public abstract HttpResponse access(URL url, Map<String, String> header,
InputStream content, Long contentLength,
        HttpMethodName httpMethod) throws Exception;

    public HttpResponse access(URL url, Map<String, String> header,
HttpMethodName httpMethod) throws Exception {
        return this.access(url, header, null, 0l, httpMethod);
    }

    public HttpResponse access(URL url, InputStream content, Long contentLength,
HttpMethodName httpMethod)
        throws Exception {
        return this.access(url, null, content, contentLength, httpMethod);
    }
    public HttpResponse access(URL url, HttpMethodName httpMethod) throws
Exception {
        return this.access(url, null, null, 0l, httpMethod);
    }

    public abstract void close();

    public String getServiceName() {
        return serviceName;
    }

    public void setServiceName(String serviceName) {
        this.serviceName = serviceName;
    }

    public String getRegion() {
        return region;
    }

    public void setRegion(String region) {
        this.region = region;
    }

    public String getAk() {
        return ak;
    }
```

```
    public void setAk(String ak) {
        this.ak = ak;
    }

    public String getSk() {
        return sk;
    }

    public void setSk(String sk) {
        this.sk = sk;
    }

}
```

AccessServiceImpl.java:

```
package com.cloud.apigateway.sdk.demo;

import java.io.IOException;
import java.io.InputStream;
import java.net.URISyntaxException;
import java.net.URL;
import java.util.HashMap;
import java.util.Map;

import javax.net.ssl.SSLContext;

import org.apache.http.Header;
import org.apache.http.HttpHeaders;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpDelete;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpHead;
import org.apache.http.client.methods.HttpPatch;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.methods.HttpPut;
import org.apache.http.client.methods.HttpRequestBase;
import org.apache.http.conn.ssl.AllowAllHostnameVerifier;
import org.apache.http.conn.ssl.SSLConnectionSocketFactory;
import org.apache.http.conn.ssl.SSLContexts;
import org.apache.http.conn.ssl.TrustSelfSignedStrategy;
import org.apache.http.entity.InputStreamEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;

import com.cloud.sdk.DefaultRequest;
import com.cloud.sdk.Request;
import com.cloud.sdk.auth.credentials.BasicCredentials;
import com.cloud.sdk.auth.signer.Signer;
import com.cloud.sdk.auth.signer.SignerFactory;
import com.cloud.sdk.http.HttpMethodName;


public class AccessServiceImpl extends AccessService {

 private CloseableHttpClient client = null;

 public AccessServiceImpl(String serviceName, String region, String ak,
   String sk) {
  super(serviceName, region, ak, sk);
 }

 /** {@inheritDoc} */

 public HttpResponse access(URL url, Map<String, String> headers,
   InputStream content, Long contentLength, HttpMethodName httpMethod)
   throws Exception {

  // Make a request for signing.
```

```
Request request = new DefaultRequest(this.serviceName);
try {
 // Set the request address.
 request.setEndpoint(url.toURI());

 String urlString = url.toString();

 String parameters = null;

  if (urlString.contains("?")) {
      parameters = urlString.substring(urlString.indexOf("?") + 1);
      Map parametersmap = new HashMap<String, String>();

      if (null != parameters && !"".equals(parameters)) {
                  String[] parameterarray = parameters.split("&");

                  for (String p : parameterarray) {
                      String key = p.split("=")[0];
                      String value = p.split("=")[1];
                      parametersmap.put(key, value);
                  }
                  request.setParameters(parametersmap);
        }
   }

} catch (URISyntaxException e) {
 // It is recommended to add logs in this place.
 e.printStackTrace();
}
// Set the request method.
request.setHttpMethod(httpMethod);
if (headers != null) {
 // Add request header information if required.
 request.setHeaders(headers);
}
// Configure the request content.
request.setContent(content);

// Select an algorithm for request signing.
Signer signer = SignerFactory.getSigner(serviceName, region);
// Sign the request, and the request will change after the signing.
signer.sign(request, new BasicCredentials(this.ak, this.sk));

// Make a request that can be sent by the HTTP client.
HttpRequestBase httpRequestBase = createRequest(url, null,
  request.getContent(), contentLength, httpMethod);
Map<String, String> requestHeaders = request.getHeaders();
// Put the header of the signed request to the new request.
for (String key : requestHeaders.keySet()) {
 if (key.equalsIgnoreCase(HttpHeaders.CONTENT_LENGTH.toString())) {
  continue;
 }
 httpRequestBase.addHeader(key, requestHeaders.get(key));
}

HttpResponse response = null;
SSLContext sslContext = SSLContexts.custom()
  .loadTrustMaterial(null, new TrustSelfSignedStrategy())
  .useTLS().build();
SSLConnectionSocketFactory sslSocketFactory = new SSLConnectionSocketFactory(
  sslContext, new AllowAllHostnameVerifier());

client = HttpClients.custom().setSSLSocketFactory(sslSocketFactory)
  .build();
// Send the request, and a response will be returned.
response = client.execute(httpRequestBase);
return response;
}
```

```
/**
 * Make a request that can be sent by the HTTP client.
 *
 * @param url
 *            specifies the API access path.
 * @param header
 *            specifies the header information to be added.
 * @param content
 *            specifies the body content to be sent in the API call.
 * @param contentLength
 *            specifies the length of the content. This parameter is optional.
 * @param httpMethod
 *            specifies the HTTP method to be used.
 * @return specifies the request that can be sent by an HTTP client.
 */
private static HttpRequestBase createRequest(URL url, Header header,
  InputStream content, Long contentLength, HttpMethodName httpMethod) {

 HttpRequestBase httpRequest;
 if (httpMethod == HttpMethodName.POST) {
  HttpPost postMethod = new HttpPost(url.toString());

  if (content != null) {
   InputStreamEntity entity = new InputStreamEntity(content,
     contentLength);
   postMethod.setEntity(entity);
  }
  httpRequest = postMethod;
 } else if (httpMethod == HttpMethodName.PUT) {
  HttpPut putMethod = new HttpPut(url.toString());
  httpRequest = putMethod;

  if (content != null) {
   InputStreamEntity entity = new InputStreamEntity(content,
     contentLength);
   putMethod.setEntity(entity);
  }
 } else if (httpMethod == HttpMethodName.PATCH) {
  HttpPatch patchMethod = new HttpPatch(url.toString());
  httpRequest = patchMethod;

  if (content != null) {
   InputStreamEntity entity = new InputStreamEntity(content,
     contentLength);
   patchMethod.setEntity(entity);
  }
 } else if (httpMethod == HttpMethodName.GET) {
  httpRequest = new HttpGet(url.toString());
 } else if (httpMethod == HttpMethodName.DELETE) {
  httpRequest = new HttpDelete(url.toString());
 } else if (httpMethod == HttpMethodName.HEAD) {
  httpRequest = new HttpHead(url.toString());
 } else {
  throw new RuntimeException("Unknown HTTP method name: "
    + httpMethod);
 }

 httpRequest.addHeader(header);
 return httpRequest;
}

@Override
public void close() {
 try {
  if (client != null) {
   client.close();
  }
 } catch (IOException e) {
  // It is recommended to add logs in this place.
```

```
   e.printStackTrace();
  }
 }

}
```

Demo.java:

```
package com.cloud.apigateway.sdk.demo;

import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.URL;

import org.apache.http.HttpResponse;

import com.cloud.sdk.http.HttpMethodName;


public class Demo {

 //replace real region
 private static final String region = "regionName";

 //replace real service name
 private static final String serviceName = "serviceName";

 public static void main(String[] args) {

  //replace real AK
  String ak = "akString";
  //replace real SK
  String sk = "skString";

  // get method
  //replace real url
  String url = "urlString";
  get(ak, sk, url);

  // post method
  //replace real url
  String postUrl = "urlString";
  //replace real body
  String postbody = "bodyString";
  post(ak, sk, postUrl, postbody);

  // put method
  //replace real body
  String putbody = "bodyString";
  //replace real url
  String putUrl = "urlString";
  put(ak, sk, putUrl, putbody);

  // delete method
  //replace real url
  String deleteUrl = "urlString";
  delete(ak, sk, deleteUrl);
 }

 public static void put(String ak, String sk, String requestUrl,
   String putBody) {

  AccessService accessService = null;
  try {
   accessService = new AccessServiceImpl(serviceName, region, ak, sk);
```

```
  URL url = new URL(requestUrl);
  HttpMethodName httpMethod = HttpMethodName.PUT;

  InputStream content = new ByteArrayInputStream(putBody.getBytes());
  HttpResponse response = accessService.access(url, content,
    (long) putBody.getBytes().length, httpMethod);

  System.out.println(response.getStatusLine().getStatusCode());


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

}

public static void patch(String ak, String sk, String requestUrl,
  String putBody) {

 AccessService accessService = null;
 try {
  accessService = new AccessServiceImpl(serviceName, region, ak, sk);
  URL url = new URL(requestUrl);
  HttpMethodName httpMethod = HttpMethodName.PATCH;
  InputStream content = new ByteArrayInputStream(putBody.getBytes());
  HttpResponse response = accessService.access(url, content,
    (long) putBody.getBytes().length, httpMethod);

  System.out.println(convertStreamToString(response.getEntity()
    .getContent()));
 } catch (Exception e) {
  e.printStackTrace();
 } finally {
  accessService.close();
 }

}

public static void delete(String ak, String sk, String requestUrl) {

 AccessService accessService = null;

 try {
  accessService = new AccessServiceImpl(serviceName, region, ak, sk);
  URL url = new URL(requestUrl);
  HttpMethodName httpMethod = HttpMethodName.DELETE;

  HttpResponse response = accessService.access(url, httpMethod);
  System.out.println(convertStreamToString(response.getEntity()
    .getContent()));
 } catch (Exception e) {
  e.printStackTrace();
 } finally {
  accessService.close();
 }

}

public static void get(String ak, String sk, String requestUrl) {

 AccessService accessService = null;

 try {
  accessService = new AccessServiceImpl(serviceName, region, ak, sk);
  URL url = new URL(requestUrl);
  HttpMethodName httpMethod = HttpMethodName.GET;
  HttpResponse response;
```

```
   response = accessService.access(url, httpMethod);
   System.out.println(convertStreamToString(response.getEntity()
     .getContent()));
  } catch (Exception e) {
   e.printStackTrace();
  } finally {
   accessService.close();
  }

 }

 public static void post(String ak, String sk, String requestUrl,
   String postbody) {

  AccessService accessService = new AccessServiceImpl(serviceName,
    region, ak, sk);
  URL url = null;
  try {
   url = new URL(requestUrl);
  } catch (MalformedURLException e) {
   e.printStackTrace();
  }
  InputStream content = new ByteArrayInputStream(postbody.getBytes());
  HttpMethodName httpMethod = HttpMethodName.POST;
  HttpResponse response;

  try {
   response = accessService.access(url, content,
     (long) postbody.getBytes().length, httpMethod);
   System.out.println(convertStreamToString(response.getEntity()
     .getContent()));
  } catch (Exception e) {
   e.printStackTrace();
  } finally {
   accessService.close();
  }
 }

 private static String convertStreamToString(InputStream is) {
  BufferedReader reader = new BufferedReader(new InputStreamReader(is));
  StringBuilder sb = new StringBuilder();

  String line = null;
  try {
   while ((line = reader.readLine()) != null) {
    sb.append(line + "\n");
   }
  } catch (IOException e) {
   e.printStackTrace();
  } finally {
   try {
    is.close();
   } catch (IOException e) {
    e.printStackTrace();
   }
  }

  return sb.toString();
 }

}
```

NOTE

1. Parameters **URI**, **AK**, **SK**, and **HTTP METHOD** are mandatory.
2. You can use the **request.addHeader()** method to add header information.

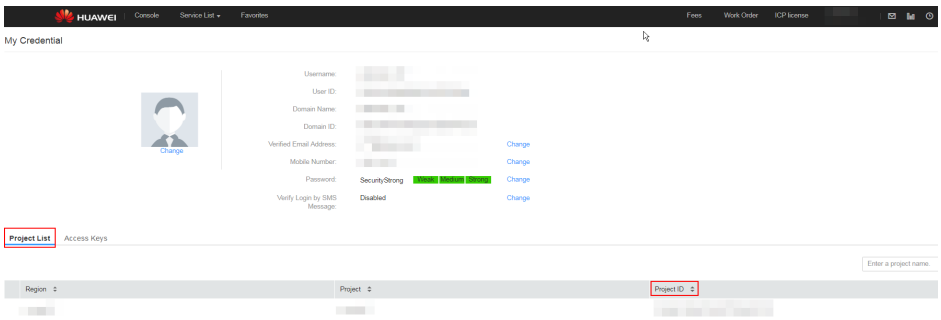
1.6 Obtaining a Project ID

A project ID is required for some URLs when an API is called. It can be **project_id** or **tenant_id** because **project_id** has the same meaning as **tenant_id** in this document. Before calling an API, you need to obtain a project ID on the console. The steps are as follows:

1. Log in to the management console.
2. Hover the mouse over the username and select **Basic Information** from the drop-down list.
3. On the **Account Info** page, click **Manage** after **Security Credentials**.

   On the **My Credential** page, view the project ID in the project list.

Figure 1-1 Viewing the project ID



2 Common Message Headers

This chapter describes common request and response REST message headers.

2.1 Common Request Headers

Table 2-1 Common request headers

Parameter	Description	Mandatory	Example Value
x-sdk-date	Specifies the time when the request is sent. The time is in **YYYYMMDD'T'HHMMSS'Z'** format. The value is the current GMT time of the system.	No This field is mandatory for AK/SK authentication.	20150907T101459Z
Authorization	Specifies the authentication information. The value can be obtained from the request signing result. For details, see section **1.5.2 Request Signing Procedure**.	No This field is mandatory for AK/SK authentication.	SDK-HMAC-SHA256 Credential=ZIRRKMTWPTQFQI1WKNKB/20150907//ec2/sdk_request, SignedHeaders=content-type;host;x-sdk-date, Signature=55741b610f3c9fa3ae40b5a8021ebf7ebc2a28a603fc62d25cb3bfe6608e1994

Parameter	Description	Mandatory	Example Value
Host	Specifies the server domain name and port number of the resources being requested. The value can be obtained from the URL of the service API. The value is *hostname[:port]*. If the port number is not specified, the default port is used. The default port number for **https** is **443**.	No This field is mandatory for AK/SK authentication.	code.test.com or code.test.com:443
Content-Type	Specifies the request body MIME type. You are advised to use the default value **application/json**. For interfaces used to upload objects or images, the value can vary depending on the flow type.	Yes	application/json
Content-Length	Specifies the length of the request body. The unit is byte.	No	3495
X-Project-Id	Specifies the project ID. Obtain the project ID by following the instructions in section **1.6 Obtaining a Project ID**. This parameter is mandatory for a request from a DeC or multi-project user.	No This field is mandatory for requests that use AK/SK authentication in the Dedicated Cloud (DeC) scenario or multi-project scenario.	e9993fc787d94b6c886cbaa340f9c0f4

Parameter	Description	Mandatory	Example Value
X-Auth-Token	Specifies the user token. For details about how to obtain the token, see section "Obtaining the User Token" in the *Identity and Access Management API Reference*. After the request is processed, the value of **X-Subject-Token** in the message header is the token value.	No This field is mandatory for token authentication.	The following is part of an example token: MIIPAgYJKoZIhvcNAQc-CoIIO8zCCDu8CAQExDTALBglghkgBZQMEAgEwgg1QBgkqhkiG9w0BBwGggg1BBIINPXsidG9rZ.

NOTE

For details about other parameters in the message header, see the HTTP protocol documentation.

2.2 Common Response Headers

Table 2-2 Common response headers

Name	Description	Example Value
Content-Length	Specifies the length of the response body. The unit is byte.	-
Date	Specifies the GMT time when a request response is returned.	Wed, 27 Dec 2016 06:49:46 GMT
Content-Type	Specifies the response body MIME type.	application/json

3 NAT Gateway Service

3.1 Overview

Object Introduction

This section describes operations for the NAT Gateway service, including creating a NAT gateway, querying a NAT gateway list, querying NAT gateway details, updating a NAT gateway, and deleting a NAT gateway.

Object Model

Table 3-1 NAT gateway object

Attribute	Type	CRUD	Default Value	Constraint	Description
id	String (UUID)	R	Automatically generated	N/A	Specifies the ID of the NAT gateway.
name	String (64 characters)	CRU	N/A	N/A	Specifies the name of the NAT gateway. **NOTE** The name can contain only digits, letters, underscores (_), hyphens (-), and Chinese.
description	String (255 characters)	CRU	N/A	N/A	Provides supplementary information about the NAT gateway.
router_id	String (UUID)	CR	N/A	The value must be an existing router ID.	Specifies the router ID used by the NAT gateway.

Attribute	Type	CRUD	Default Value	Constraint	Description
internal_network_id	String (UUID)	CR	N/A	The value must be an existing network ID, and the network type can be only VXLAN.	Specifies the network ID used by the NAT gateway.
status	String (16 characters)	R	N/A	N/A	Specifies the status of the NAT gateway. The value can be **ACTIVE**, **ERROR**, **PENDING_CREATE**, **PENDING_UPDATE**, or **PENDING_DELETE**.
spec	String (16 characters)	CRU	N/A	N/A	Specifies the specifications of the NAT gateway. The value can be **1**, **2**, **3**, or **4**. **1**: small scale **2**: medium scale **3**: large scale **4**: super-large scale
tenant_id	String (255 characters)	CR	N/A	N/A	Specifies the tenant ID.
created_at	Date	R	None	The value must be a date.	Specifies the date when the NAT gateway is created.
admin_state_up	Bool	RU	True	The value can be only **True** or **False**.	Specifies the administrator status. Only the system administrator has the operation rights. **NOTE** **True** indicates that the NAT gateway has been unfrozen. **False** indicates that the NAT gateway has been frozen.

3.2 Creating a NAT Gateway

Function

This API is used to create a NAT gateway.

API Format

Method	URI	Description
POST	/v2.0/nat_gateways	Creates a NAT gateway.

Restrictions

None

Request Parameters

Parameter	Type	Mandatory	Description
tenant_id	String	No	Specifies the tenant ID.
name	String(64)	Yes	Specifies the name of the NAT gateway. **NOTE** The name can contain only digits, letters, underscores (_), hyphens (-), and Chinese.
description	String(255)	No	Provides supplementary information about the NAT gateway.
spec	String	Yes	Specifies the specifications of the NAT gateway. The value can be **1**, **2**, **3**, or **4**. ● **1**: small scale ● **2**: medium scale ● **3**: large scale ● **4**: extra-large scale
router_id	String	Yes	Specifies the router UUID.
internal_network_id	String	Yes	Specifies the downstream interface of the NAT gateway, that is, the next hop of the DVR.

Response Parameters

Parameter	Type	Mandatory	Description
nat_gateway	Dict	Yes	Specifies the NAT gateway object list. For details, see **Table 3-1**.

Example Request

```
POST /v2.0/nat_gateways
{
    "nat_gateway": {
        "name": "nat_001",
        "description": "my nat gateway 01",
        "router_id": "d84f345c-80a1-4fa2-a39c-d0d397c3f09a",
        "internal_network_id": "89d66639-aacb-4929-969d-07080b0f9fd9",
        "spec": "1"
    }
}
```

Example Response

```
{
    "nat_gateway": {
        "router_id": "d84f345c-80a1-4fa2-a39c-d0d397c3f09a",
        "status": "PENDING_CREATE",
        "description": "my nat gateway 01",
        "admin_state_up": true,
        "tenant_id": "27e25061336f4af590faeabeb7fcd9a3",
        "created_at": "2017-11-18 07:34:32.203044",
        "spec": "1",
        "internal_network_id": "89d66639-aacb-4929-969d-07080b0f9fd9",
        "id": "a78fb3eb-1654-4710-8742-3fc49d5f04f8",
        "name": "nat_001"
    }
}
```

Response Codes

For details, see **A Response Codes**.

3.3 Querying a NAT Gateway List

Function

This API is used to query a NAT gateway list.

API Format

Method	URI	Description
GET	/v2.0/nat_gateways?id={id}&name={name }&description={description}&router_id={router_id}&internal_network_id={ internal_network_id }&admin_state_up={admin_state_up}&tenant_id={tenant_id}&spec={spec}&status={status}&created_at={created_at}&admin_state_up={admin_state_up}	Queries all NAT gateways available to the tenant submitting the request.

Restrictions

None

Request Parameters

None

Response Parameters

Parameter	Type	Mandatory	Description
nat_gateways	List (NAT gateways)	Yes	Specifies the NAT gateway object list. For details, see **Table 3-1**.

Example Request

```
GET /v2.0/nat_gateways
```

Example Response

```
{
    "nat_gateways": [
        {
            "router_id": "b1d81744-5165-48b8-916e-e56626feb88f",
            "status": "ACTIVE",
            "description": "",
            "admin_state_up": true,
            "tenant_id": "27e25061336f4af590faeabeb7fcd9a3",
            "created_at": "2017-11-15 14:50:39.505112",
            "spec": "2",
            "internal_network_id": "5930796a-6026-4d8b-8790-6c6bfc9f87e8",
            "id": "a253be25-ae7c-4013-978b-3c0785eccd63",
            "name": "wj3"
        },
        {
            "router_id": "305dc52f-13dd-429b-a2d4-444a1039ba0b",
            "status": "ACTIVE",
            "description": "",
            "admin_state_up": true,
            "tenant_id": "27e25061336f4af590faeabeb7fcd9a3",
            "created_at": "2017-11-17 07:41:07.538062",
            "spec": "2",
```

```
            "internal_network_id": "fc09463b-4ef8-4c7a-93c8-92d9ca6daf9d",
            "id": "e824f1b4-4290-4ebc-8322-cfff370dbd1e",
            "name": "lyl001"
        }
    ]
}
```

Response Codes

For details, see **A Response Codes**.

3.4 Querying Details About a Specified NAT Gateway

Function

This API is used to query details about a specified NAT gateway.

API Format

Method	URI	Description
GET	/v2.0/nat_gateways/{nat_gateway_id}	Queries details about a specified NAT gateway.

Restrictions

None

Request Parameters

None

Response Parameters

Parameter	Type	Mandatory	Description
nat_gateway	Dict	Yes	Specifies the NAT gateway object list. For details, see **Table 3-1**.

Example Request

```
GET /v2.0/nat_gateways/a78fb3eb-1654-4710-8742-3fc49d5f04f8
```

Example Response

```
{
    "nat_gateway": {
        "router_id": "d84f345c-80a1-4fa2-a39c-d0d397c3f09a",
        "status": "ACTIVE",
        "description": "my nat gateway 01",
        "admin_state_up": true,
        "tenant_id": "27e25061336f4af590faeabeb7fcd9a3",
```

```
        "created_at": "2017-11-18 07:34:32.203044",
        "spec": "1",
        "internal_network_id": "89d66639-aacb-4929-969d-07080b0f9fd9",
        "id": "a78fb3eb-1654-4710-8742-3fc49d5f04f8",
        "name": "nat_001"
    }
}
```

Response Codes

For details, see **A Response Codes**.

3.5 Updating a NAT Gateway

Function

This API is used to update a NAT gateway.

API Format

Method	URI	Description
PUT	/v2.0/nat_gateways/{ nat_gateway_id }	Updates a NAT gateway.

Restrictions

You can update the NAT gateway name only when the **admin_state_up** parameter is set to **True** and **status** to **ACTIVE**.

You can update the NAT gateway description only when the **admin_state_up** parameter is set to **True** and **status** to **ACTIVE**.

You can update the NAT gateway specifications only when the **admin_state_up** parameter is set to **True** and **status** to **ACTIVE**.

Request Parameters

Parameter	Type	Mandatory	Description
nat_gateway	Dict	Yes	Specifies the NAT gateway object list. For details, see **Table 3-1**. Mandatory field: None. Only the **name**, **description**, and **spec** fields can be updated. At least one attribute must be specified for the NAT gateway to be updated.

Response Parameters

Parameter	Type	Mandatory	Description
nat_gateway	Dict	Yes	Specifies the NAT gateway object list. For details, see **Table 3-1**.

Example Request

```
PUT /v2.0/nat_gateways/a78fb3eb-1654-4710-8742-3fc49d5f04f8
  {
    "nat_gateway": {
        "name": "new_name",
        "description": "new description",
        "spec": "1"
    }
}
```

Example Response

```
{
    "nat_gateway": {
        "router_id": "d84f345c-80a1-4fa2-a39c-d0d397c3f09a",
        "status": "ACTIVE",
        "description": "new description",
        "admin_state_up": true,
        "tenant_id": "27e25061336f4af590faeabeb7fcd9a3",
        "created_at": "2017-11-18 07:34:32.203044",
        "spec": "1",
        "internal_network_id": "89d66639-aacb-4929-969d-07080b0f9fd9",
        "id": "a78fb3eb-1654-4710-8742-3fc49d5f04f8",
        "name": "new_name"
    }
}
```

Response Codes

For details, see **A Response Codes**.

3.6 Deleting a NAT Gateway

Function

This API is used to delete a NAT gateway.

API Format

Method	URI	Description
DELETE	/v2.0/nat_gateways/{ nat_gateway_id }	Deletes a NAT gateway.

Request Parameters

None

Response Parameters

None

Example Request

```
DELETE /v2.0/nat_gateways/a78fb3eb-1654-4710-8742-3fc49d5f04f8
```

Example Response

```
None (Status code 204)
```

Response Codes

For details, see **A Response Codes**.

4 SNAT Rules

4.1 Overview

Object Introduction

This section describes operations for SNAT rules, including creating an SNAT rule, querying an SNAT rule list, query SNAT rule details, and deleting an SNAT rule.

Object Model

Table 4-1 SNAT rule object

Attribute	Type	CRUD	Default Value	Constraint	Description
id	String (UUID)	R	Automatically generated	N/A	Specifies the ID of the SNAT rule.
tenant_id	String (255 characters)	CR	N/A	N/A	Specifies the tenant ID.
nat_gateway_id	String (UUID)	CR	N/A	N/A	Specifies the ID of the NAT gateway.
network_id	String (UUID)	CR	N/A	The value must be an existing network ID, and the network type can be only VXLAN.	Specifies network ID used by the SNAT rule.

Attribute	Type	CRUD	Default Value	Constraint	Description
floating_ip_id	String (UUID)	CR	N/A	The value must be an existing floating IP address.	Specifies the ID of the floating IP address.
floating_ip_address	String (64 characters)	R	N/A	The value must be a valid floating IP address.	Specifies the floating IP address.
status	String (16 characters)	R	N/A	N/A	Specifies the status of the SNAT rule. The value can be **ACTIVE**, **ERROR**, **PENDING_CREATE**, **PENDING_UPDATE**, or **PENDING_DELETE**.
created_at	Date	R	N/A	The value must be a date.	Specifies the date when the SNAT rule is created.
admin_state_up	Bool	RU	True	The value can be only **True** or **False**.	Specifies the administrator status. Only the system administrator has the operation rights. **NOTE** **True** indicates that the SNAT rule has been unfrozen. **False** indicates that the SNAT rule has been frozen.

4.2 Creating an SNAT Rule

Function

This API is used to create an SNAT rule.

API Format

Method	URI	Description
POST	/v2.0/snat_rules	Creates an SNAT rule.

Restrictions

You can create an SNAT rule only when **status** of the NAT gateway is set to **ACTIVE** and **admin_state_up** of the NAT gateway administrator to **True**.

Request Parameters

Parameter	Type	Mandatory	Description
nat_gateway_id	String	Yes	Specifies the ID of the NAT gateway.
network_id	String	Yes	Specifies the network ID.
floating_ip_id	String	Yes	Specifies the ID of the floating IP address.

Response Parameters

Parameter	Type	Mandatory	Description
snat_rule	Dict	Yes	Specifies the SNAT rule object list. For details, see **Table 4-1**.

Example Request

```
POST /v2.0/snat_rules
{
    "snat_rule": {
        "nat_gateway_id": "a78fb3eb-1654-4710-8742-3fc49d5f04f8",
        "network_id": "eaad9cd6-2372-4be1-9535-9bd37210ae7b",
        "floating_ip_id": "bdc10a4c-d81a-41ec-adf7-de857f7c812a"
    }
}
```

Example Response

```
{
    "snat_rule": {
        "floating_ip_id": "bdc10a4c-d81a-41ec-adf7-de857f7c812a",
        "status": "PENDING_CREATE",
        "nat_gateway_id": "a78fb3eb-1654-4710-8742-3fc49d5f04f8",
        "admin_state_up": true,
        "network_id": "eaad9cd6-2372-4be1-9535-9bd37210ae7b",
        "tenant_id": "27e25061336f4af590faeabeb7fcd9a3",
        "created_at": "2017-11-18 07:54:21.665430",
        "id": "5b95c675-69c2-4656-ba06-58ff72e1d338",
        "floating_ip_address": "5.21.11.226"
```

```
    }
}
```

Response Codes

For details, see **A Response Codes**.

4.3 Querying an SNAT Rule List

Function

This API is used to query an SNAT rule list.

API Format

Method	URI	Description
GET	/v2.0/snat_rules?id={id}&tenant_id={tenant_id}&nat_gateway_id={ nat_gateway_id }&network_id={network_id}&floating_ip_id={floating_ip_id}&floating_ip_address={ floating_ip_address}&status={status}&created_at={created_at}&Admin_state_up={Admin_state_up}	Queries all SNAT rules available to the tenant submitting the request.

Restrictions

None

Request Parameters

None

Response Parameters

Parameter	Type	Mandatory	Description
snat_rules	List (SNAT rules)	Yes	Specifies the SNAT rule object list. For details, see **Table 4-1**.

Example Request

```
GET /v2.0/snat_rules/
```

Example Response

```
{
    "snat_rules": [
      {
```

```
            "floating_ip_id": "bf99c679-9f41-4dac-8513-9c9228e713e1",
            "status": "ACTIVE",
            "nat_gateway_id": "cda3a125-2406-456c-a11f-598e10578541",
            "admin_state_up": true,
            "network_id": "9a469561-daac-4c94-88f5-39366e5ea193",
            "tenant_id": "abc",
            "created_at": "2017-11-15 15:44:42.595173",
            "id": "79195d50-0271-41f1-bded-4c089b2502ff",
            "floating_ip_address": "5.21.11.242"
        },
        {
            "floating_ip_id": "6e496fba-abe9-4f5e-9406-2ad8c809ac8c",
            "status": "ACTIVE",
            "nat_gateway_id": "e824f1b4-4290-4ebc-8322-cfff370dbd1e",
            "admin_state_up": true,
            "network_id": "97e89905-f9c8-4ae3-9856-392b0b2fbe7f",
            "tenant_id": "abc",
            "created_at": "2017-11-17 07:43:44.830845",
            "id": "4a1a10d7-0d9f-4846-8cda-24cffeffef5c",
            "floating_ip_address": "5.21.11.142"
        }
    ]
}
```


Response Codes

For details, see **A Response Codes**.

4.4 Querying Details About a Specified SNAT Rule

Function

This API is used to query details about a specified SNAT rule.

API Format

Method	URI	Description
GET	/v2.0/snat_rules/{snat_rule_id}	Queries details about a specified SNAT rule.

Restrictions

None

Request Parameters

None

Response Parameters

Parameter	Type	Mandatory	Description
snat_rule	Dict	Yes	Specifies the SNAT rule object list. For details, see **Table 4-1**.

Example Request

```
GET /v2.0/snat_rules/5b95c675-69c2-4656-ba06-58ff72e1d338
```

Example Response

```
{
    "snat_rule": {
        "floating_ip_id": "bdc10a4c-d81a-41ec-adf7-de857f7c812a",
        "status": "ACTIVE",
        "nat_gateway_id": "a78fb3eb-1654-4710-8742-3fc49d5f04f8",
        "admin_state_up": true,
        "network_id": "eaad9cd6-2372-4be1-9535-9bd37210ae7b",
        "tenant_id": "27e25061336f4af590faeabeb7fcd9a3",
        "created_at": "2017-11-18 07:54:21.665430",
        "id": "5b95c675-69c2-4656-ba06-58ff72e1d338",
        "floating_ip_address": "5.21.11.226"
    }
}
```

Response Codes

For details, see **A Response Codes**.

4.5 Deleting an SNAT Rule

Function

This API is used to delete an SNAT rule.

API Format

Method	URI	Description
DELETE	/v2.0/snat_rules/{snat_rule_id}	Deletes an SNAT rule.

Request Parameters

None

Response Parameters

None

Example Request

```
DELETE /v2.0/snat_rules/a78fb3eb-1654-4710-8742-3fc49d5f04f8
```

Example Response

```
None (Status code 204)
```

Response Codes

For details, see **A Response Codes**.

5 DNAT Rules

5.1 Overview

Object Introduction

This section describes operations for DNAT rules, including creating a DNAT rule, querying a DNAT rule list, query DNAT rule details, and deleting a DNAT rule.

Object Model

Table 5-1 DNAT rule object

Attribute	Type	CRUD	Default Value	Constraint	Description
id	Uuid-Str	R	Automatically generated	N/A	Specifies the ID of the DNAT rule.
tenant_id	String(255)	CR	N/A	N/A	Specifies the tenant ID.
nat_gateway_id	Uuid-Str	CR	N/A	N/A	Specifies the ID of the NAT gateway.
port_id	Uuid-Str	CR	N/A	The port cannot be the one used by network devices.	Specifies the port ID of an ECS or a BMS.

Attribute	Type	CRUD	Default Value	Constraint	Description
internal_service_port	Int	CR	N/A	1-65535	Specifies port used by ECSs or BMSs to provide services for external systems.
floating_ip_id	Uuid-Str	CR	N/A	N/A	Specifies the ID of the floating IP address.
external_service_port	Int	CR	N/A	1-65535	Specifies the port bound to the floating IP address.
floating_ip_address	String(64)	R	N/A	The value must be a valid floating IP address.	Specifies the floating IP address.
protocol	Str or Int	CR	N/A	N/A	Currently, the value can be a character string, for example, **TCP** or **UDP**, or a specific protocol number, such as 6 or 17. Querying DNAT rules by protocol number is not supported.

Attribute	Type	CRUD	Default Value	Constraint	Description
status	String(16)	R	N/A	N/A	Specifies the status of the NAT gateway. The value can be **ACTIVE**, **ERROR**, **PENDING_CREATE**, or **PENDING_UPDATE**. PENDING_DELETE
created_at	DATE	R	N/A	DATE	Specifies the date when the DNAT rule is created.
admin_state_up	Bool	RU	True	The value can be only **True** or **False**.	Specifies the administrator status. Only the system administrator has the operation rights. **NOTE** **True** indicates that the DNAT rule has been unfrozen. **False** indicates that the DNAT rule has been frozen.

5.2 Creating a DNAT Rule

Function

This API is used to create a DNAT rule.

API Format

Method	URI	Description
POST	/v2.0/dnat_rules	Creates a DNAT rule

Restrictions

You can create a DNAT rule only when **status** of the NAT gateway is set to **ACTIVE** and **admin_state_up** of the NAT gateway administrator to **True**.

Request Parameters

Parameter	Type	Mandatory	Description
nat_gateway_id	String	Yes	Specifies the ID of the NAT gateway.
port_id	String	Yes	Specifies the port ID of a VM or bare-metal server.
internal_service_port	Int	Yes	Specifies port used by VMs or bare-metal servers to provide services for external systems.
floating_ip_id	String	Yes	Specifies the ID of the floating IP address.
external_service_port	Int	Yes	Specifies the port for providing external services.
protocol	Str or Int	Yes	Specifies the protocol type. Currently, TCP and UDP are supported. The protocol number of TCP and UDP is 6 and 17, respectively.

Response Parameters

Parameter	Type	Mandatory	Description
dnat_rule	Dict	Yes	Specifies the DNAT rule object list. For details, see **Table 5-1**.

Example Request

```
POST /v2.0/dnat_rules
{
    "dnat_rule": {
            "floating_ip_id": "bf99c679-9f41-4dac-8513-9c9228e713e1",
            "nat_gateway_id": "cda3a125-2406-456c-a11f-598e10578541",
            "port_id": "9a469561-daac-4c94-88f5-39366e5ea193",
            "internal_service_port": 993,
            "protocol": "tcp",
            "external_service_port": 242
    }
}
```

Example Response

```
{
   "dnat_rule":
      {
         "floating_ip_id": "bf99c679-9f41-4dac-8513-9c9228e713e1",
         "status": "ACTIVE",
         "nat_gateway_id": "cda3a125-2406-456c-a11f-598e10578541",
         "admin_state_up": true,
         "port_id": "9a469561-daac-4c94-88f5-39366e5ea193",
"internal_service_port": 993,
         "protocol": "TCP",
         "tenant_id": "abc",
         "created_at": "2017-11-15 15:44:42.595173",                    "id":
"79195d50-0271-41f1-bded-4c089b2502ff",
         "floating_ip_address": "5.21.11.226"
         "external_service_port": 242
     }
}
```

Response Codes

For details, see **A Response Codes**.

5.3 Querying a DNAT Rule List

Function

This API is used to query a DNAT rule list.

API Format

Method	URI	Description
GET	/v2.0/dnat_rules?id={id}&tenant_id={tenant_id}&port_id={ port_id }&internal_service_port={internal_service_port}&floating_ip_id={floating_ip_id}&floating_ip_address={floating_ip_address}&external_service_port={external_service_port }&protocol=protocol&status={status}&created_at={created_at}&admin_state_up={admin_state_up}	Queries all DNAT rules available to the tenant submitting the request.

Restrictions

None

Request Parameters

None

Response Parameters

Parameter	Type	Mandatory	Description
dnat_rules	List(dnat_rule)	Yes	Specifies the DNAT rule object list. For details, see **Table 5-1**.

Example Request

```
GET /v2.0/dnat_rules/
```

Example Response

```
{
    "dnat_rules": [
      {
            "floating_ip_id": "bf99c679-9f41-4dac-8513-9c9228e713e1",
            "status": "ACTIVE",
            "nat_gateway_id": "cda3a125-2406-456c-a11f-598e10578541",
            "admin_state_up": true,
            "port_id": "9a469561-daac-4c94-88f5-39366e5ea193",
            "internal_service_port": 993,
            "protocol": "tcp",
            "tenant_id": "abc",
            "created_at": "2017-11-15 15:44:42.595173",
            "id": "79195d50-0271-41f1-bded-4c089b2502ff",
            "floating_ip_address": "5.21.11.226",
            "external_service_port": 242
        },
      {
            "floating_ip_id": "Cf99c679-9f41-4dac-8513-9c9228e713e1",
            "status": "ACTIVE",
            "nat_gateway_id": "Dda3a125-2406-456c-a11f-598e10578541",
            "admin_state_up": true,
```

```
            "port_id": "Aa469561-daac-4c94-88f5-39366e5ea193",
            "internal_service_port": 993,
            "protocol": "udp",
            "tenant_id": "abc",
            "created_at": "2017-11-15 15:44:42.595173",
            "id": "79195d50-0271-41f1-bded-4c089b2502ff",
            "floating_ip_address": "5.21.11.226",
            "external_service_port": 242
        }
    ]
}
```

Response Codes

For details, see **A Response Codes**.

5.4 Querying Details About a Specified DNAT Rule

Function

This API is used to query details about a specified DNAT rule.

API Format

Method	URI	Description
GET	/v2.0/dnat_rules/{dnat_rule_id}	Queries details about a specified DNAT rule.

Restrictions

None

Request Parameters

None

Response Parameters

Parameter	Type	Mandatory	Description
dnat_rule	Dict	Yes	Specifies the DNAT rule object list. For details, see **Table 5-1**.

Example Request

```
GET /v2.0/dnat_rules/5b95c675-69c2-4656-ba06-58ff72e1d338
```

Example Response

```
{
    {
```

```
        "floating_ip_id": "Cf99c679-9f41-4dac-8513-9c9228e713e1",
"status": "ACTIVE",
        "nat_gateway_id": "Dda3a125-2406-456c-a11f-598e10578541",
"admin_state_up": true,
        "port_id": "Aa469561-daac-4c94-88f5-39366e5ea193",
"internal_service_port": 993,
        "protocol": "TCP",
        "tenant_id": "abc",
        "created_at": "2017-11-15 15:44:42.595173",
        "id": "79195d50-0271-41f1-bded-4c089b2502ff",
        "floating_ip_address": "5.21.11.226"
        "external_service_port": 242
    }
}
```

Response Codes

For details, see **A Response Codes**.

5.5 Deleting a DNAT Rule

Function

This API is used to delete a DNAT rule.

API Format

Method	URI	Description
DELETE	/v2.0/dnat_rules/{dnat_rule_id}	Deletes a DNAT rule.

Request Parameters

None

Response Parameters

None

Example Request

```
DELETE /v2.0/dnat_rules/a78fb3eb-1654-4710-8742-3fc49d5f04f8
```

Example Response

```
None (Status code 204)
```

Response Codes

For details, see **A Response Codes**.

A Response Codes

Normal Response Code	Type	Description
200	OK	Specifies the normal response code for the GET and PUT operations.
201	Created	Specifies the normal response code for the POST operation.
204	No Content	Specifies the normal response code for the DELETE operation.

Error Response Code	Description
400 Bad Request	The server failed to process the request.
401 Unauthorized	You must enter a username and the password to access the requested page.
403 Forbidden	You are forbidden to access the requested page.
404 Not Found	The server could not find the requested page.
405 Method Not Allowed	You are not allowed to use the method specified in the request.
406 Not Acceptable	The response generated by the server could not be accepted by the client.
407 Proxy Authentication Required	You must use the proxy server for authentication so that the request can be processed.
408 Request Timeout	The request timed out.
409 Conflict	The request could not be processed due to a conflict.

Error Response Code	Description
500 Internal Server Error	Failed to complete the request because of an internal service error.
501 Not Implemented	Failed to complete the request because the server does not support the requested function.
502 Bad Gateway	Failed to complete the request because the server has received an invalid response.
503 Service Unavailable	Failed to complete the request because the service is unavailable.
504 Gateway Timeout	A gateway timeout error occurred.

B Change History

Date	Description
2018-03-30	This issue is the third official release, which incorporates the following change: Added section **5 DNAT Rules**.
2018-01-30	This issue is the second official release, which incorporates the following change: Modified the description of the **name** attribute in section **3.1 Overview**.
2017-12-18	This issue is the first official release.