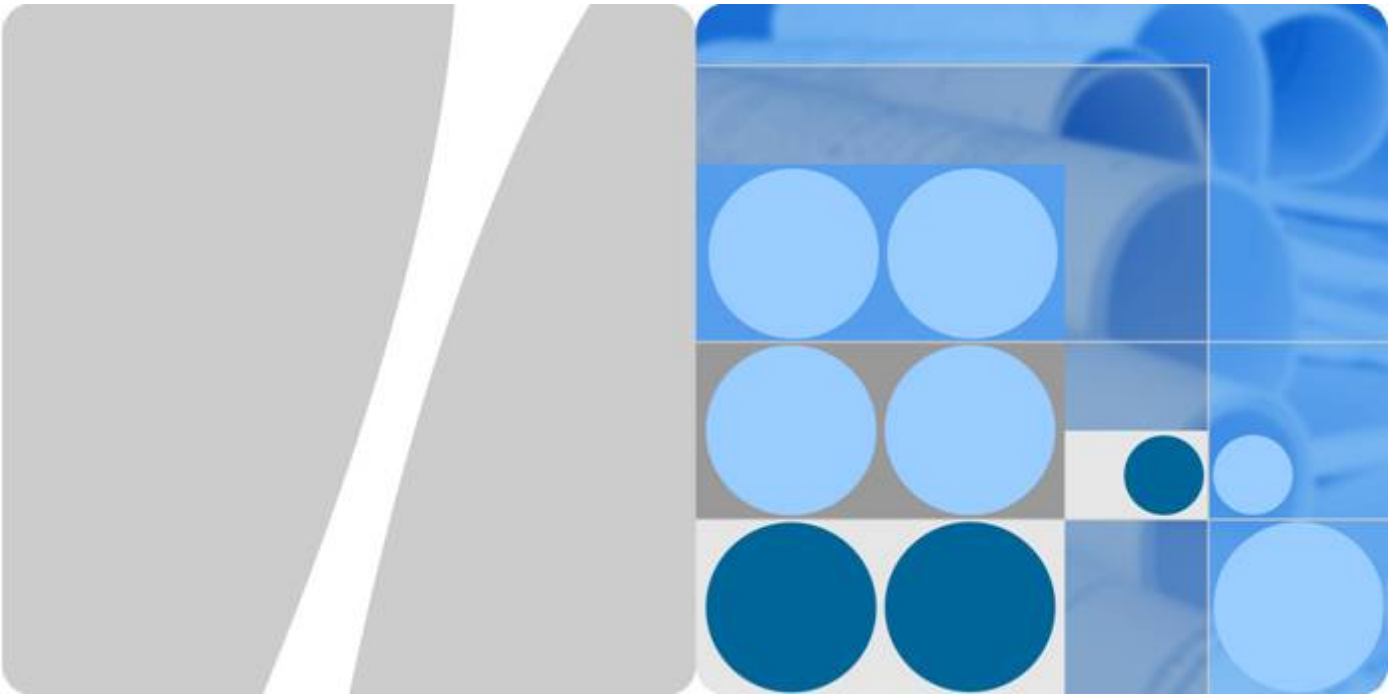

统一身份认证服务

API 参考

文档版本　06

发布日期　2016-12-30

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： http://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

目 录

1 接口调用方法

第三方应用对公有云API的访问需经过签名认证。

第三方应用对FusionCloud Stack API的访问需经过签名认证。

本章主要介绍了使用签名的过程和注意事项，并通过示例代码展示了如何使用默认的Signer对请求进行签名和利用HTTP Client发送请求。

1.1 服务使用方法

公有云FusionCloud Stack API符合REST API的设计理论。

REST从资源的角度来观察整个网络，分布在各处的资源由URI（Uniform Resource Identifier）确定，而客户端的应用通过URL（Unified Resource Locator）来获取资源。

URL的一般格式为（带方括号的为可选项）：

https://Endpoint [/uri]

URL中的参数说明，如**表1-1**所示。

表 1-1 URL 中的参数说明

参数	描述	是否必选
protocol	请求使用的协议类型，例如https。 https表示通过安全的SSL加密的HTTP协议访问该资源。	必选
Endpoint	存放资源的服务器的域名，从**地区和终端节点**“地区和终端节点”中获取。	必选
uri	资源路径，也即API访问路径。从每个接口的URI段落中获取，例如：获取用户token接口的URI为“v3/auth/tokens”。	可选

1.2 请求方法

在HTTP协议中，可以使用多种请求方法，用于指明以何种方式来访问指定的资源,例如：GET、 PUT、 POST、 DELETE、PATCH。目前提供的REST接口支持的请求方法如下表所示。

表 1-2 请求方法一览表

方法	说明
GET	请求服务器返回指定资源。
PUT	请求服务器更新指定资源。
POST	请求服务器新增资源或执行特殊操作。
DELETE	请求服务器删除指定资源，如删除对象等。
PATCH	请求服务器更新资源的部分内容。 当资源不存在的时候，PATCH可能会去创建一个新的资源。

1.3 请求认证方式

调用接口有如下两种认证方式，您可以任选其中一种进行认证鉴权。

- Token认证：通过Token认证调用请求。
- AK/SK认证：通过AK(Access Key ID)/SK(Secret Access Key)加密调用请求。AK/SK认证安全性更高。

1.4 Token 认证

获取 Token 认证步骤

当您使用Token认证方式时，请采用如下步骤调用接口：

步骤1 发送“POST https://***IAM的Endpoint***/v3/auth/tokens”，获取IAM的Endpoint及消息体中的区域名称，请参考**地区和终端节点**“地区和终端节点”。

步骤2 获取Token，请参考“**3.1 获取用户token**”。

说明

- Token值即接口返回消息Header中的“X-Subject-Token”的值。
- Token的有效期为24小时，需要同一个Token鉴权时，可以先缓存起来，避免频繁调用。

步骤3 调用接口时，在业务接口请求消息头中增加“X-Auth-Token”，取值为上面步骤中的Token值。

----结束

获取 Token 认证示例

步骤1 发送“POST https://***API网关地址***/v3/auth/tokens”，请求内容示例如下：

说明

下面示例代码中的斜体字需要替换为实际内容，详情请参考“**3.1 获取用户token**”。

```
{
  "auth": {
    "identity": {
      "methods": [
        "password"
      ],
      "password": {
        "user": {
          "name": "username",
          "password": "password",
          "domain": {
            "name": "domainname"
          }
        }
      }
    },
    "scope": {
      "project": {
        "name": "eu-de-01" //For example, the region name is eu-de-01.
      }
```

```
        }
    }
}
```

步骤2 请求响应成功后在响应消息头中包含的“X-Subject-Token”的值即为Token值。

步骤3 调用业务接口，在请求消息头中增加“X-Auth-Token”，“X-Auth-Token”的取值为前面步骤中获取的Token。

----结束

1.5 AK/SK 认证

通过API网关向下层服务发送请求时，必须使用AK(Access Key ID)、SK(Secret Access Key)对请求进行签名。

说明

AK：访问密钥ID。与私有访问密钥关联的唯一标识符；访问密钥ID和私有访问密钥一起使用，对请求进行加密签名。

SK：与访问密钥ID结合使用的密钥，对请求进行加密签名，可标识发送方，并防止请求被修改。

1.5.1 生成 AK/SK

步骤1 注册并登录管理控制台，单击右上方登录的账号，选择“我的认证”。

步骤2 在“我的认证”页面，选择“管理访问密钥”页签。

步骤3 单击列表下侧的“新增访问密钥”，创建新的访问密钥。

说明

每个用户最多可创建2个有效访问密钥（AK/SK），且一旦生成永久有效，为了账户安全性，建议定期更换访问密钥。

步骤4 输入当前用户的登录密码，并通过邮箱或者手机进行验证，输入对应的验证码。

说明

在统一身份认证服务中创建的用户，如果创建时未填写邮箱或者手机号，则只需校验登录密码。

步骤5 单击“确定”，下载访问密钥。

说明

为防止访问密钥泄露，建议您将其保存到安全的位置。

----结束

1.5.2 请求签名流程

签名前的准备

1. 下载API网关签名工具。

   下载地址：**http://esdk.huawei.com/ilink/esdk/download/HW_456706**

   下载地址：**https://oss.prod-cloud-ocb.orange-business.com/download/develop/api/java/java-sdk-core.zip**

2. 解压下载的压缩包。

3. 创建java工程，将解压出来的jar引用到依赖路径中。

签名过程

步骤1 创建用于签名的请求com.cloud.sdk.DefaultRequest(JAVA)。

步骤2 设置DefaultRequest的目标API URL、HTTPS方法、内容等信息。

步骤3 对DefaultRequest进行签名：

1. 调用SignerFactory.getSigner(String serviceName, String regionName)获取一个签名工具实现的实例。
2. 调用Signer.sign(Request<?> request, Credentials credentials)对步骤1创建的请求进行签名。

   以下代码展示了这个步骤：

```
 //选用签名算法，对请求进行签名
Signer signer = SignerFactory.getSigner(serviceName, region);
//对请求进行签名，request会发生改变
signer.sign(request, new BasicCredentials(this.ak, this.sk));
```

步骤4 把步骤3签名产生的request转换为一个适合发送的请求，并将签名后request中的header信息放入新的request中。

以Apache HttpClient为例，需要把DefaultRequest转换为HttpRequestBase，把签名后的DefaultRequest的header信息放入HttpRequestBase中。

具体过程请查看“1.5.3 示例代码”中的AccessServiceImpl.java。

----结束

1.5.3 示例代码

下面代码展示了如何对一个请求进行签名，并通过HTTP Client发送一个HTTPS请求的过程：

代码分成三个类进行演示：

- AccessService:抽象类，将GET/POST/PUT/DELETE归一成access方法。
- Demo:运行入口，模拟用户进行GET/POST/PUT/DELETE请求。
- AccessServiceImpl:实现access方法，具体与API网关通信的代码都在access方法中。

下面示例代码中的region和serviceName（英文服务名缩写），请从地区和终端节点“地区和终端节点”中获取。

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();
 }

}
```

说明

1. URI、AK、SK、HTTP METHOD是必须设置的参数。
2. 可通过request.addHeader()添加头信息。

1.6 获取项目 ID

在调用接口的时候，部分URL中需要填入项目ID（project_id或者tenant_id，本文中project_id和tenant_id含义一样），所以需要在“我的认证”页面获取项目ID。

项目ID获取步骤如下：

步骤1 注册并登录管理控制台。

步骤2 单击登录账户并选择“我的认证”。

步骤3 在“我的认证”页面的“项目列表”中查看项目ID。

----结束

2 公共消息头

REST公共消息头包含两类：公共请求消息头和公共响应消息头，本章对这部分内容进行介绍。

2.1 公共请求消息头

表 2-1 公共请求消息头

参数	描述	是否必选	示例
x-sdk-date	请求的发生时间，格式为(YYYYMMDD'T'HHMMSS'Z')。 取值为当前系统的GMT时间。	否 使用AK/SK认证时必选。	20150907T101459Z
Authorization	签名认证信息。 该值来源于请求签名结果。 请参考**1.5.2 请求签名流程**。	否 使用AK/SK认证时必选。	SDK-HMAC-SHA256 Credential=ZIRRKMTWPTQFQI1WKNKB/20150907//ec2/sdk_request, SignedHeaders=content-type;host;x-sdk-date, Signature=55741b610f3c9fa3ae40b5a8021ebf7ebc2a28a603fc62d25cb3bfe6608e1994
Host	请求的服务器信息，从服务API的URL中获取。值为hostname[:port]。未写端口时使用默认的端口，https的默认端口为443。	否 使用AK/SK认证时必选。	code.test.com 或者 code.test.com:443
Content-type	发送的实体的MIME类型。	是	application/json
Content-Length	请求消息体长度，单位为Byte。	POST/PUT请求必填。GET不能包含。	3495
X-Project-Id	project id，用于获取不同project的token。	否	e9993fc787d94b6c886cbaa340f9c0f4
X-Auth-Token	用户token。	否 使用token认证时必选。	-

说明

其它header属性，请遵照http协议。

2.2 公共响应消息头

表 2-2 公共响应消息头

参数	描述
Content-Length	响应消息体的字节长度，单位为Byte。
Date	系统响应的时间。
Content-type	发送的实体的MIME类型。

3 API 参考

IAM通过如下安全协议和行为来保证API接口的安全性：

1. 所有API接口都使用HTTPS协议。
2. 首先，客户端使用正确的凭证从服务端获取token；然后，客户端在调用服务端API的同时会将token发送给服务端；最后，服务端通过校验客户端token的有效性来决定是否为客户端提供服务。
3. API接口提供鉴权能力，因此拥有相应权限的的客户端才能成功调用对应的API。

3.1 获取用户 token

功能介绍

该接口用来获取用户token，可以通过用户名/密码的方式进行认证。

说明

- token的有效期为24小时，需要同一个token鉴权时，可以先缓存起来，避免频繁调用。
- 该接口属于服务间接口，提供了锁定机制用于防止暴力破解，服务调用时，请做好用户名密码的充分校验。

URI

- URI格式

  POST /v3/auth/tokens

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。

- Request Body参数说明

参数	是否为必选	类型	说明
methods	是	String Array	该字段内容为“password”。
user	是	Json Object	示例： `"user": { "name": "username", "password": "password", "domain": { "name": "domainname" } }` *domainname*：用户所属的企业账户名称。 *username*：用户名称。 *password*：用户登录时的密码。

参数	是否为必选	类型	说明

scope	否	Json Object	token的作用范围。支持domain、project，但两者不能设置在同一层级上。
			示例1（“name”为企业账户名称，此处以“examplename”为例）：
			`"scope": { "domain": { "name": "examplename" } }`
			该示例表示本token仅能访问examplename企业账户下的资源。
			示例2：
			`"scope": { "project": { "id": "0215ef11e49d4743be23dd97a1561e91" } }`
			该示例表示本token仅能访问所属企业账户下的id为0215ef11e49d4743be23dd97a1561e91的project下的资源。
			示例3：
			`"scope": { "project": { "domain": { "name": "examplename" }, "name": "project_example" } }`
			该示例表示本token仅能访问examplename企业账户下名称为project_example的project下的资源。

说明

上述参数信息的获取方式请参考“**3.32 相关参数信息获取**”。

- 请求样例

获取用户名为Lucy，登录密码为Examplepassword123，所在domain名为examplename的token。

```
{
  "auth": {
    "identity": {
      "methods": ["password"],
      "password": {
        "user": {
          "name": "Lucy",
          "password": "Examplepassword123",
          "domain": {
            "name": "examplename"
          }
        }
```

```
      }
    },
    "scope": {
      "domain": {
        "name": "examplename"
      }
    }
  }
}
```

响应

- Response Header参数说明

参数	类型	描述
X-Subject-Token	String	签名后的token。

- Token格式说明

参数	是否为必选	类型	描述
methods	是	String	获取token的方式。
expires_at	是	String	token超时时间。
issued_at	是	String	token产生时间。
user	是	Json Object	示例： `"user": { "name": "username", "id": "userid", "domain": { "name": "domainname", "id": "domainid" } }` username：用户名称。 userid：用户id。 domainname：用户所属的企业账户名称。 domainid：用户所属的企业账户的域id。
domain	否	Json Object	根据请求中的scope，判断是否返回该字段。 示例： `"domain": { "name" : "domainame", "id" : "domainid"}` domainname：企业账户名称。 domainid：企业账户的域id。

参数	是否为必选	类型	描述
project	否	Json Object	根据请求中的scope，判断是否返回该字段。 示例： `"project": { "name": "projectname", "id": "projectid", "domain": { "name": "domainname", "id": "domainid" } }` projectname：project名称。 projectid：project的id。 domainname：poject所属的企业账户名称。 domainid：project所属的企业账户的域id。
roles	是	Json Object	角色数组。 示例： `"roles" : [{ "name" : "role1", "id" : "roleid1" }, { "name" : "role2", "id" : "roleid2" } ]`

- 响应样例

```
Response Header中存储信息为：
X-Subject-
Token:MIIDkgYJKoZIhvcNAQcCoIIDgzCCA38CAQExDTALBglghkgBZQMEAgEwgXXXXX...

Response Body中存储信息为：
{
  "token" : {
    "methods" : ["password"],
    "expires_at" : "2015-11-09T01:42:57.527363Z",
    "issued_at" : "2015-11-09T00:42:57.527404Z",
    "user" : {
      "domain" : {
      "id" : "default",
      "name" : "Default"
      },
      "id" : "ee4dfb6e5540447cb3741905149d9b6e",
      "name" : "admin"
    },
    "domain" : {
       "name" : "Default",
       "id" : "default"
    },
    "roles" : [{
       "name" : "role1",
       "id" : "roleid1"
       }, {
       "name" : "role2",
       "id" : "roleid2"
       }
```

```
  ]
  }
}
```

状态码

状态码	说明
201	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.2 查询指定 project 的信息

功能介绍

该接口用于查询指定project信息。

URI

- URI格式

  GET /v3/projects{?domain_id,name,page,per_page}

- 参数说明

参数	是否为必选	类型	说明
domain_id	否	String	用户所属企业账户的id。
name	否	String	project的名称。
page	否	Integer	查询第几页的数据，查询值最小为1。
per_page	否	Integer	每页的数据个数，取值范围为[1,5000]。

说明

- 该接口至少需要一种查询条件。禁止全量查询project列表。
- 需要分页查询时，必须保证查询参数中同时存在page和per_page。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	已认证的token。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -X "X-Auth-Token:$token" -X GET https://10.145.93.56:31943/v3/projects?domain_id=5c9f5525d9d24c5bbf91e74d86772029&name=Region002
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
projects	是	List	project列表。

- projects格式说明

参数	是否为必选	类型	说明
description	是	String	project描述。
enabled	是	Boolean	project是否可用。
id	是	String	project id。
domain_id	是	String	project所在domain的id。
name	是	String	project名称。

- 响应样例

```
{
  "links": {
    "self": "https://www.example.com/v3/projects?domain_id=c9f5525d9d24c5bbf91e74d86772029&name=test9",
    "previous": null,
    "next": null
  },
  "projects": [
    {
      "description": "",
      "links": {
        "self": "https://www.example.com/v3/projects/e86737682ab64b2490c48f08bcc41914"
      },
      "enabled": true,
      "id": "0215ef11e49d4743be23dd97a1561e91",
      "domain_id": "c9f5525d9d24c5bbf91e74d86772029",
      "name": "test9"
    }
  ]
}
```

状态码

状态码	说明
200	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.3 校验指定 token 有效性

功能介绍

该接口用于校验指定token的有效性，如果有效则返回token的详细信息。

URI

- URI格式

GET /v3/auth/tokens

请求

- Request Header参数说明

参数	是否为必选	类型	说明
X-Auth-Token	是	String	已认证的拥有secu_admin权限的token。 **说明** 用户校验自己的token有效性时，不需要鉴权。
X-Subject-Token	是	String	待校验的token。该token和X-Auth-Token属于同一租户。

- 查询参数说明

参数	是否为必选	类型	说明
nocatalog	否	String	token中不显示catalog信息。

- 请求样例

```
curl -i -k -H "X-Auth-Token:$token" -H "X-Subject-Token:$token" -X GET https://
172.30.48.86:31943/v3/auth/tokens
```

响应

- Response Header参数说明

参数	是否为必选	类型	说明
X-Auth-Token	是	String	已认证的拥有secu_admin权限的token。
X-Subject-Token	是	String	待校验的token。该token和X-Auth-Token属于同一租户。

- Response body参数说明

参数	类型	描述
token	Object	token信息列表。

- token格式说明

参数	是否为必选	类型	描述
methods	是	Array	获取token的方式。
expires_at	是	String	token超时时间。
issued_at	是	String	token产生时间。

user	是	Object	示例：
			"user": {
			"name": "username",
			"id": "userid",
			"domain": {
			"name": "domainname",
			"id": "domainid"
			}
			}
			username：用户名称。
			userid：用户id。
			domainname：用户所属的企业账户名称。
			domainid：用户所属的企业账户的域id。

参数	是否为必选	类型	描述
domain	否	Object	根据请求中的scope，判断是否返回该字段。 示例： `"domain": { "name" : "domainame", "id" : "domainid"}` domainname：企业账户名称。 domainid：企业账户的域id。
project	否	Object	根据请求中的scope，判断是否返回该字段。 示例： `"project": { "name": "projectname", "id": "projectid", "domain": { "name": "domainname", "id": "domainid" } }` projectname：project名称。 projectid：project的id。 domainname：poject所属的企业账户名称。 domainid：project所属的企业账户的域id。
roles	是	Array	角色数组。 示例： `"roles" : [{ "name" : "role1", "id" : "roleid1" }, { "name" : "role2", "id" : "roleid2" } ]`

- Response样例

```
{
  "token" : {
    "methods" : ["password"],
    "expires_at" : "2015-11-09T01:42:57.527363Z",
    "issued_at" : "2015-11-09T00:42:57.527404Z",
    "user" : {
      "domain" : {
      "id" : "default",
      "name" : "Default"
      },
      "id" : "ee4dfb6e5540447cb3741905149d9b6e",
      "name" : "admin"
    },
    "domain" : {
       "name" : "Default",
       "id" : "default"
    },
    "roles" : [{
```

```
        "name" : "role1",
        "id" : "roleid1"
        }, {
        "name" : "role2",
        "id" : "roleid2"
        }
    ]
    }
}
```

状态码

状态码	说明
200	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
503 Service Unavailable	服务不可用。

3.4 查询 Keystone API 版本信息

功能介绍

该接口用于获取Keystone API的版本信息。

URI

- URI格式

GET /v3

请求

- 请求样例

```
curl -i -k -X GET https://172.30.48.86:31943/v3
```

响应

- 响应参数说明

参数	类型	描述
version	Object	Keystone API的版本信息。

- version格式说明

参数	是否为必选	类型	说明
status	是	string	版本状态。
updated	是	string	版本最后更新时间。
media-types	是	Array	版本支持的消息格式。
id	是	string	版本号，如v3.0。
links	是	Array	版本的链接。

响应样例（响应成功）

```
{
  "version": {
    "status": "stable",
    "updated": "2013-03-06T00:00:00Z",
    "media-types": [
      {
        "base": "application/json",
        "type": "application/vnd.openstack.identity-v3+json"
      },
      {
        "base": "application/xml",
        "type": "application/vnd.openstack.identity-v3+xml"
      }
    ],
    "id": "v3.0",
    "links": [
      {
        "href": "https://172.30.48.86:31943/identity/v3/",
        "rel": "self"
      }
    ]
  }
}
```

状态码

状态码	说明
200	请求成功。
400 Bad Request	请求错误。
404 Not Found	找不到资源。
503 Service Unavailable	服务不可用。

3.5 查询 Services

功能介绍

该接口用于查询服务列表。

URI

- URI格式

GET /v3/services{?type}

- 参数说明

参数	是否为必选	类型	说明
type	否	String	服务类型，类型的值可以是compute,ec2, identity, image, network, 或volume。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	已认证的token。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X GET https://172.30.48.86:31943/v3/services?type=compute
```

响应

- Reponse Body参数说明

参数	是否为必选	类型	说明
links	是	Dict	服务的链接，包含next,previous,self。
services	是	List	服务列表。

- services格式说明

参数	是否为必选	类型	说明
description	是	String	服务描述。
enabled	是	Boolean	服务是否可用。

参数	是否为必选	类型	说明
id	是	String	服务id。
name	否	String	服务名。
type	是	String	服务类型。
links	是	Dict	服务的链接。

响应样例（响应成功）

```
{
    "services": [
        {
            "name": "compute5",
            "links": {
                "self": "https://iamcore_links.com/v3/services/053d21d488d1463c818132d9d08fb617"
            },
            "enabled": true,
            "type": "compute",
            "id": "053d21d488d1463c818132d9d08fb617",
            "description": "Compute service 5"
        },
        {
            "name": "compute3",
            "links": {
                "self": "https://iamcore_links.com/v3/services/c2474183dca7453bbd73123a0b78feae"
            },
            "enabled": true,
            "type": "compute",
            "id": "c2474183dca7453bbd73123a0b78feae",
            "description": "Compute service 3"
        },
        {
            "name": "compute2",
            "links": {
                "self": "https://iamcore_links.com/v3/services/c7166694ebdd4616bd927737f7b12ca2"
            },
            "enabled": true,
            "type": "compute",
            "id": "c7166694ebdd4616bd927737f7b12ca2",
            "description": "Compute service 2"
        }
    ],
    "links": {
        "self": "https://iamcore_links.com/v3/services?type=compute",
        "previous": null,
        "next": null
    }
}
```

状态码

状态码	说明
200	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。

状态码	说明
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.6 查询 Endpoints

功能介绍

该接口用于查询终端地址列表，提供服务访问入口。

URI

- URI格式

GET /v3/endpoints{? interface, service_id}

- 参数说明

参数	是否为必选	类型	说明
interface	否	String	过滤终端地址平面，类型取值可以是public，internal或admin。
service_id	否	String	过滤服务id。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	已认证的token。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X GET https://172.30.48.86:31943/v3/endpoints?
interface=public&service_id=43cbe5e77aaf4665bbb962062dc1fc9d
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
links	是	dict	终端地址的链接，包含next，previous，self。
endpoints	是	list	终端地址列表。

- endpoints格式说明

参数	是否为必选	类型	说明
id	是	String	终端地址id。
url	是	String	终端地址的地址。
region	是	String	终端地址的区域。
region_id	是	String	终端地址的区域id。
enabled	是	Boolean	终端地址是否可用。
interface	是	String	终端地址的平面。
service_id	是	String	终端地址所属服务的id。
links	是	dict	终端地址的链接。

响应样例（请求成功）

```
{
    "endpoints": [
        {
            "region_id": null,
            "links": {
                "self": "https://iamcore_links.com/v3/endpoints/162277d696f54cf592f19b569f85d158"
            },
            "url": "https://10.185.190.104:7443/v2/$(tenant_id)s",
            "region": null,
            "enabled": true,
            "interface": "public",
            "service_id": "053d21d488d1463c818132d9d08fb617",
            "id": "162277d696f54cf592f19b569f85d158"
        }
    ],
    "links": {
        "self": "https://iamcore_links.com/v3/endpoints?
service_id=053d21d488d1463c818132d9d08fb617&interface=public",
        "previous": null,
```

```
        "next": null
    }
}
```

状态码

状态码	说明
200	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.7 查询用户详情

功能介绍

该接口用于查询指定用户的详细信息。

URI

- URI格式

  GET /v3/users/{user_id}
- 参数说明

参数	是否为必选	类型	说明
user_id	是	String	用户ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。

参数	是否为必选	类型	说明
X-Auth-Token	是	String	已认证的拥有Security Administrator权限的token，或用户自身的token。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X GET https://172.30.48.86:31943/v3/users/43cbe5e77aaf4665bbb962062dc1fc9d
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
user	是	Object	user对象。

- user格式说明

参数	是否为必选	类型	说明
enabled	是	Boolean	是否启用用户。
id	是	String	用户ID。
domain_id	是	String	用户所在domain的id。
name	是	String	用户名称。
links	是	Object	用户资源的链接。
default_project_id	否	String	用户默认的project ID。
password_expires_at	是	String	密码过期时间（UTC时间），null表示密码不过期。

- 响应样例

```
{
    "user": {
        "default_project_id": "263fd9",
        "domain_id": "1789d1",
        "enabled": true,
        "id": "9fe1d3",
        "links": {
            "self": "https://example.com/identity/v3/users/9fe1d3"
        },
        "name": "jsmith",
        "password_expires_at": "2016-11-06T15:32:17.000000"
```

```
    }
}
```

状态码

状态码	说明
200	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.8 创建用户

功能介绍

该接口用于在某一租户下创建用户。

URI

- URI格式
  POST /v3/users

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	已认证的拥有Security Administrator权限的token。

- Request Body参数说明

参数	是否为必选	类型	说明
name	是	String	用户名，长度5~32之间，只能包含大写字母、小写字母、空格、数字、特殊字符（-_）且不能以数字开头。
domain_id	否	String	用户所在domain的ID。
enabled	否	Boolean	是否启用用户，true为启用，false为停用，默认为true。
password	否	String	用户密码。 ● 长度在8~32之间； ● 包含以下四种字符中的三种： 大写字母、小写字母、数字和特殊字符； ● 必须满足账户设置中密码策略的要求。
default_project_id	否	String	用户默认的project ID。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X POST -d'{"user": {"default_project_id": "acf2ffabba974fae8f30378ffde2cfa6","domain_id": "88b16b6440684467b8825d7d96e154d8","enabled": true,"name": "jamesdoe","password": "********"}}' https://172.30.48.86:31943/v3/users
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
user	是	Object	user对象。

- user格式说明

参数	是否为必选	类型	说明
enabled	是	Boolean	是否启用用户，true为启用，false为停用，默认为true。
id	是	String	用户ID。
domain_id	是	String	用户所在domain的id。
name	是	String	用户名称。
links	是	Object	用户资源的链接。
default_project_id	否	String	用户默认的project ID。

- 响应样例

```
{
  "user": {
    "name": "jamesdoe",
    "links": {
      "self": "https://iam.sa-brazil-1.telefonicaopencloud.com/v3/users/
614d1d2fb86940faab8f350bf1b9dbac"
    },
    "domain_id": "88b16b6440684467b8825d7d96e154d8",
    "enabled": true,
    "id": "614d1d2fb86940faab8f350bf1b9dbac",
    "default_project_id": "acf2ffabba974fae8f30378ffde2cfa6"
  }
}
```

状态码

状态码	说明
201	创建成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
409 Conflict	资源冲突。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.9 修改用户信息

功能介绍

该接口用于修改租户下对应的用户信息。

URI

- URI格式

  PATCH /v3/users/{user_id}

- URI参数说明

参数	是否为必选	类型	说明
user_id	是	String	用户ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	已认证的拥有Security Administrator权限的token。

- Request Body参数说明

参数	是否为必选	类型	说明
name	否	String	用户名，长度5~32之间，只能包含大写字母、小写字母、空格、数字、特殊字符（-_）且不能以数字开头。
domain_id	否	String	用户所在domain的ID。
enabled	否	Boolean	是否启用用户，true为启用，false为停用，默认为true。
password	否	String	修改后的用户密码。 ● 长度在8~32之间； ● 包含以下四种字符中的三种： 大写字母、小写字母、数字和特殊字符； ● 不能包含手机号和邮箱； ● 必须满足账户设置中密码策略的要求。
default_project_id	否	String	用户默认的project ID。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H
"X-Auth-Token:$token" -X PATCH -d'{"user": {"name":"james1234","default_project_id":
"88b16b6440684467b8825d7d96e154d8","enabled": false,"password":"********"}}' https://
172.30.48.86:31943/v3/users/2c1c6c54e59141b889c99e6fada5f19f
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
user	是	Object	user对象。

- user格式说明

参数	是否为必选	类型	说明
enabled	是	Boolean	是否启用用户，ture为启用，false为停用，默认为true。
id	是	String	用户ID。
domain_id	是	String	用户所在domain的id。
name	是	String	用户名称。
links	是	Object	用户资源的链接。
default_project_id	否	String	用户默认的project ID。
password_expires_at	是	String	密码过期时间（UTC时间），null表示密码不过期。

- 响应样例

```
{
  "user": {
    "name": "james1234",
    "links": {
      "self": "https://iam.sa-brazil-1.telefonicaopencloud.com/v3/users/
6d8b04e3bf99445b8f76300903e5bf32"
    },
    "extra": {},
    "domain_id": "88b16b6440684467b8825d7d96e154d8",
    "enabled": false,
    "id": "6d8b04e3bf99445b8f76300903e5bf32",
    "default_project_id": "88b16b6440684467b8825d7d96e154d8",
    "password_expires_at": "2016-12-07T00:00:00.000000Z"
  }
}
```

状态码

状态码	说明
200	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
409 Conflict	资源冲突。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。

状态码	说明
503 Service Unavailable	服务不可用。

3.10 删除用户

功能介绍

该接口用于删除指定用户。

URI

- URI格式

  DELETE /v3/users/{user_id}
- URI参数说明

参数	是否为必选	类型	说明
user_id	是	String	用户ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	已认证的拥有Security Administrator权限的token。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X DELETE https://172.30.48.86:31943/v3/users/2c1c6c54e59141b889c99e6fada5f19f
```

状态码

状态码	说明
204	删除成功。
400 Bad Request	请求错误。

状态码	说明
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.11 查询用户列表

功能介绍

该接口用于查询用户列表。

URI

- URI格式
  GET /v3/users
- 参数说明

参数	是否为必选	类型	说明
domain_id	否	String	用户所属的domain ID。
enabled	否	String	是否启用用户，true为启用，false为停用，默认为true。
name	否	String	用户名。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。

参数	是否为必选	类型	说明
X-Auth-Token	是	String	已认证的拥有Security Administrator权限的token。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X GET https://172.30.48.86:31943/v3/users
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
users	是	Array	user对象数组。
links	是	Object	资源访问链接。

- user格式说明

参数	是否为必选	类型	说明
enabled	是	Boolean	是否启用用户，true为启用，false为停用，默认为true。
id	是	String	用户ID。
domain_id	是	String	用户所在domain的id。
name	是	String	用户名。
links	是	Object	用户资源的链接。
default_project_id	否	String	用户默认的project ID。
password_expires_at	是	Strng	密码过期时间（UTC时间），null表示密码不过期。

- 响应样例

```
{
  "users": [
    {
      "name": "james1234",
      "links": {
        "self": "https://iam.sa-brazil-1.telefonicaopencloud.com/v3/users/6d8b04e3bf99445b8f76300903e5bf32"
```

```
      },
      "domain_id": "88b16b6440684467b8825d7d96e154d8",
      "enabled": false,
      "id": "6d8b04e3bf99445b8f76300903e5bf32",
      "default_project_id": "88b16b6440684467b8825d7d96e154d8",
      "password_expires_at": "2016-12-07T00:00:00.000000Z"
    }
  ],
  "links": {
    "self": "https://iam.sa-brazil-1.telefonicaopencloud.com/v3/users?
domain_id=88b16b6440684467b8825d7d96e154d8&enabled=false",
    "previous": null,
    "next": null
  }
}
```

状态码

状态码	说明
200	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.12 修改密码

功能介绍

该接口用于用户修改自己密码。

URI

- URI格式

  POST /v3/users/{user_id}/password

- URI参数说明

参数	是否为必选	类型	说明
user_id	是	String	用户ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	用户自身的token。

- Request Body参数说明

参数	是否为必选	类型	说明
original_password	是	String	用户的原密码。
password	是	String	用户的新密码。 ● 长度8~32位字符； ● 包含以下四种字符的三种： 大写字母、小写字母、数字和特殊字符； ● 不能包含手机号和邮箱； ● 必须满足账户设置中密码策略的要求。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X POST -d'{"user": {"password": "********", "original_password": "********"}}' https://172.30.48.86:31943/v3/users/2c1c6c54e59141b889c99e6fada5f19f/password
```

状态码

状态码	说明
204	修改成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。

状态码	说明
503 Service Unavailable	服务不可用。

3.13 查询用户所属用户组

功能介绍

该接口用于查询指定用户所属的用户组信息。

URI

- URI格式

  GET /v3/users/{user_id}/groups
- 参数说明

参数	是否为必选	类型	说明
user_id	是	String	用户ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	已认证的拥有Security Administrator权限的token。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X GET https://172.30.48.86:31943/v3/users/43cbe5e77aaf4665bbb962062dc1fc9d/groups
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
groups	是	Array	用户组列表。
links	是	Object	资源访问链接。

- group格式说明

参数	是否为必选	类型	说明
description	是	String	用户组描述。
id	是	String	用户组ID。
domain_id	是	String	用户组所在domain的id。
name	是	String	用户组名称。
links	是	Object	用户组资源访问链接。

- 响应样例

```
{
  "links": {
    "self": "https://iam.sa-brazil-1.telefonicaopencloud.com/v3/users/
f7cb4876e5174c0885433e280e831c43/groups",
    "previous": null,
    "next": null
  },
  "groups": [
    {
      "description": "User group that has the permission for all system operations",
      "links": {
        "self": "https://iam.sa-brazil-1.telefonicaopencloud.com/v3/groups/
e21c7a1e415c4604927948dc24750716"
      },
      "id": "e21c7a1e415c4604927948dc24750716",
      "create_time": 1472888495993,
      "domain_id": "88b16b6440684467b8825d7d96e154d8",
      "name": "admin"
    }
  ]
}
```

状态码

状态码	说明
200	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.14 查询用户 project 列表

功能介绍

该接口用于查询指定用户可访问的project信息。

URI

- URI格式

  GET /v3/users/{user_id}/projects

- 参数说明

参数	是否为必选	类型	说明
user_id	是	String	用户ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	已认证的拥有Security Administrator权限的token，或用户自身的token。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X GET https://172.30.48.86:31943/v3/users/43cbe5e77aaf4665bbb962062dc1fc9d/projects
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
projects	是	Array	project列表。
links	是	Object	资源访问链接。

- project格式说明

参数	是否为必选	类型	说明
description	是	String	project描述。
id	是	String	project ID。
domain_id	是	String	project所在domain的id。
name	是	String	project名称。
links	是	Object	project资源访问链接。

- 响应样例

```
{
  "links": {
    "self": "https://iam.sa-brazil-1.telefonicaopencloud.com/v3/users/
f7cb4876e5174c0885433e280e831c43/projects",
    "previous": null,
    "next": null
  },
  "projects": [
    {
      "description": "",
      "links": {
        "self": "https://iam.sa-brazil-1.telefonicaopencloud.com/v3/projects/
19f2bff8bc6c413c84e1fc31d0a389c3"
      },
      "enabled": true,
      "id": "19f2bff8bc6c413c84e1fc31d0a389c3",
      "domain_id": "88b16b6440684467b8825d7d96e154d8",
      "name": "MOS"
    },
    {
      "description": "",
      "links": {
        "self": "https://iam.sa-brazil-1.telefonicaopencloud.com/v3/projects/
34bf38accc4d4c4f828ff5a708818023"
      },
      "enabled": true,
      "id": "34bf38accc4d4c4f828ff5a708818023",
      "domain_id": "88b16b6440684467b8825d7d96e154d8",
      "name": "de-de"
    }
  ]
}
```

状态码

状态码	说明
200	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。

状态码	说明
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.15 注册 Identity Provider

功能介绍

该接口用于注册一个Identity Provider。

URI

- URI格式

PUT /v3/OS-FEDERATION/identity_providers/{id}

- 参数说明

参数	是否为必选	类型	说明
id	是	String	Identity Provider的ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	具有Security Administrator权限token。

- Request Body参数说明

参数	是否为必选	类型	说明
description	否	String	Identity Provider的描述信息。

参数	是否为必选	类型	说明
enabled	否	Boolean	Identity Provider是否启用，true为启用，false为停用，默认为false。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X PUT -d'{"identity_provider":{"description":"Stores ACME identities.","enabled":true}}' https://10.185.190.118:31943/v3/OS-FEDERATION/identity_providers/ACME
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
id	是	String	Identity Provider的ID。
description	是	String	Identity Provider的描述信息。
enabled	是	Boolean	Identity Provider是否启用。
links	是	Object	Identity Provider的资源链接，包含protocols，self。

响应样例

```
{
    "identity_provider": {
        "description": "Stores ACME identities",
        "enabled": true,
        "id": "ACME",
        "links": {
            "protocols": "https://example.com/v3/OS-FEDERATION/identity_providers/ACME/protocols",
            "self": "https://example.com/v3/OS-FEDERATION/identity_providers/ACME"
        }
    }
}
```

状态码

状态码	说明
201	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。

状态码	说明
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.16 查询 Identity Provider 列表

功能介绍

该接口用于查询Identity Provider列表信息。

URI

- URI格式

GET /v3/OS-FEDERATION/identity_providers

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	所属租户的token。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X GET https://10.185.190.118:31943/v3/OS-FEDERATION/identity_providers
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
identity_providers	是	List of objects	Identity Provider的列表。

参数	是否为必选	类型	说明
links	是	Object	Identity Provider的资源链接，包含next，previous，self。

响应样例

```
{
    "identity_providers": [
        {
            "description": "Stores ACME identities",
            "enabled": true,
            "id": "ACME",
            "links": {
                "protocols": "https://example.com/v3/OS-FEDERATION/identity_providers/ACME/
protocols",
                "self": "https://example.com/v3/OS-FEDERATION/identity_providers/ACME"
            }
        },
        {
            "description": "Stores contractor identities",
            "enabled": false,
            "id": "ACME-contractors",
            "links": {
                "protocols": "https://example.com/v3/OS-FEDERATION/identity_providers/ACME-
contractors/protocols",
                "self": "https://example.com/v3/OS-FEDERATION/identity_providers/ACME-contractors"
            }
        }
    ],
    "links": {
        "next": null,
        "previous": null,
        "self": "https://example.com/v3/OS-FEDERATION/identity_providers"
    }
}
```

状态码

状态码	说明
200	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.17 查询 Identity Provider

功能介绍

该接口用于查询Identity Provider信息。

URI

- URI格式

GET /v3/OS-FEDERATION/identity_providers/{id}

- 参数说明

参数	是否为必选	类型	说明
id	是	String	Identity Provider的ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	所属租户的token。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X GET https://10.185.190.118:31943/v3/OS-FEDERATION/identity_providers/ACME
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
id	是	String	Identity Provider的ID。
description	是	String	Identity Provider的描述信息。

参数	是否为必选	类型	说明
enabled	是	Boolean	Identity Provider是否启用，true为启用，false为停用，默认为false。
links	是	Object	Identity Provider的资源链接，包含protocols，self。

响应样例

```
{
    "identity_provider": {
        "description": "Stores ACME identities",
        "enabled": false,
        "id": "ACME",
        "links": {
            "protocols": "https://example.com/v3/OS-FEDERATION/identity_providers/ACME/protocols",
            "self": "https://example.com/v3/OS-FEDERATION/identity_providers/ACME"
        }
    }
}
```

状态码

状态码	说明
200	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.18 更新 Identity Provider

功能介绍

该接口用于更新Identity Provider信息。

URI

- URI格式

PATCH /v3/OS-FEDERATION/identity_providers/{id}

- 参数说明

参数	是否为必选	类型	说明
id	是	String	Identity Provider的ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	具有Security Administrator权限的token。

- Request Body参数说明

参数	是否为必选	类型	说明
description	否	String	Identity Provider的描述信息。
enabled	否	Boolean	Identity Provider是否启用，true为启用，false为停用，默认为false。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X PATCH -d'{"identity_provider":{"enabled":false}}' https://
10.185.190.118:31943/v3/OS-FEDERATION/identity_providers/ACME
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
id	是	String	Identity Provider的ID。

参数	是否为必选	类型	说明
description	是	String	Identity Provider的描述信息。
enabled	是	Boolean	Identity Provider是否启用。
links	是	Object	Identity Provider的资源链接，包含protocols，self。

响应样例

```
{
    "identity_provider": {
        "description": "Stores ACME identities",
        "enabled": false,
        "id": "ACME",
        "links": {
            "protocols": "https://example.com/v3/OS-FEDERATION/identity_providers/ACME/protocols",
            "self": "https://example.com/v3/OS-FEDERATION/identity_providers/ACME"
        }
    }
}
```

状态码

状态码	说明
200	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
409 Conflict	资源冲突。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.19 删除 Identity Provider

功能介绍

该接口用于删除Identity Provider信息。

URI

- URI格式

DELETE /v3/OS-FEDERATION/identity_providers/{id}

- 参数说明

参数	是否为必选	类型	说明
id	是	String	Identity Provider的ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	具有Security Administrator权限的token。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X DELETE https://10.185.190.118:31943/v3/OS-FEDERATION/identity_providers/ACME
```

状态码

状态码	说明
204	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。

状态码	说明
405 Method Not Allowed	不允许的方法。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.20 注册 Mapping

功能介绍

该接口用于注册一个Mapping。

URI

- URI格式

PUT /v3/OS-FEDERATION/mappings/{id}

- 参数说明

参数	是否为必选	类型	说明
id	是	String	Mapping的ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	为已认证用户创建Mapping时所需的Security Administrator权限token。

- Request Body参数说明

参数	是否为必选	类型	说明
rules	是	String	将远端用户映射为本地用户的规则列表。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H
"X-Auth-Token:$token" -X PUT -d'{"mapping":{"rules":[{"local":[{"user":{"name":"{0}"}},
{"group":{"name":"0cd5e9"}}],"remote":[{"type":"UserName"},
{"type":"orgPersonType","not_any_of":["Contractor","Guest"]}]}]}}' https://
10.185.190.118:31943/v3/OS-FEDERATION/mappings/ACME
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
id	是	String	Mapping的ID。
rules	是	String	将远端用户映射为本地用户的规则列表字符串。
links	是	Object	Mapping的资源链接，包含self。

响应样例

```
{
    "mapping": {
        "id": "ACME",
        "links": {
            "self": "https://example.com/v3/OS-FEDERATION/mappings/ACME"
        },
        "rules": "[
            {
                "local": [
                    {
                        "user": {
                            "name": "{0}"
                        }
                    },
                    {
                        "group": {
                            "name": "0cd5e9"
                        }
                    }
                ],
                "remote": [
                    {
                        "type": "UserName"
                    },
                    {
                        "type": "orgPersonType",
                        "not_any_of": [
                            "Contractor",
                            "Guest"
                        ]
                    }
                ]
```

```
            }
        ]"
    }
}
```

状态码

状态码	说明
201	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.21 查询 Mapping 列表

功能介绍

该接口用于查询Mapping列表信息。

URI

- URI格式

GET /v3/OS-FEDERATION/mappings

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	查询Mapping列表时所属租户的token。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X GET https://10.185.190.118:31943/v3/OS-FEDERATION/mappings
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
mappings	是	List of objects	Mapping的列表。
links	是	Object	Mapping的资源链接，包含next，previous，self。

响应样例

```
{
    "links": {
        "next": null,
        "previous": null,
        "self": "https://example.com/v3/OS-FEDERATION/mappings"
    },
    "mappings": [
        {
            "id": "ACME",
            "links": {
                "self": "https://example.com/v3/OS-FEDERATION/mappings/ACME"
            },
            "rules": [
                {
                    "local": [
                        {
                            "user": {
                                "name": "{0}"
                            }
                        },
                        {
                            "group": {
                                "id": "0cd5e9"
                            }
                        }
                    ],
                    "remote": [
                        {
                            "type": "UserName"
                        },
                        {
                            "type": "orgPersonType",
                            "any_one_of": [
                                "Contractor",
                                "SubContractor"
                            ]
                        }
                    ]
                }
            ]
        }
    ]
}
```

状态码

状态码	说明
200	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.22 查询 Mapping

功能介绍

该接口用于查询Mapping信息。

URI

- URI格式

GET /v3/OS-FEDERATION/mappings/{id}

- 参数说明

参数	是否为必选	类型	说明
id	是	String	Mapping的ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。

参数	是否为必选	类型	说明
X-Auth-Token	是	String	查询Mapping时所属租户的token。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X GET https://10.185.190.118:31943/v3/OS-FEDERATION/mappings/ACME
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
id	是	String	Mapping的ID。
rules	是	String	将远端用户映射为本地用户的规则列表。
links	是	Object	Mapping的资源链接，包含self。

响应样例

```
{
    "mapping": {
        "id": "ACME",
        "links": {
            "self": "https://example.com/v3/OS-FEDERATION/mappings/ACME"
        },
        "rules": [
            {
                "local": [
                    {
                        "user": {
                            "name": "{0}"
                        }
                    },
                    {
                        "group": {
                            "name": "0cd5e9"
                        }
                    }
                ],
                "remote": [
                    {
                        "type": "UserName"
                    },
                    {
                        "type": "orgPersonType",
                        "not_any_of": [
                            "Contractor",
                            "Guest"
                        ]
                    }
                ]
            }
        ]
    }
}
```

状态码

状态码	说明
200	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.23 更新 Mapping

功能介绍

该接口用于更新Mapping信息。

URI

- URI格式

PATCH /v3/OS-FEDERATION/mappings/{id}

- 参数说明

参数	是否为必选	类型	说明
id	是	String	Mapping的ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。

参数	是否为必选	类型	说明
X-Auth-Token	是	String	更新已认证用户名下的Mapping时所需的Security Administrator权限token。

- Request Body参数说明

参数	是否为必选	类型	说明
rules	是	String	将远端用户映射为本地用户的规则列表字符串。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H
"X-Auth-Token:$token" -X PATCH -d'{"mapping":{"rules":[{"local":[{"user":{"name":"{0}"}},
{"group":{"name":"0cd5e9"}}],"remote":[{"type":"UserName"},
{"type":"orgPersonType","any_one_of":["Contractor","SubContractor"]}]}]}}' https://
10.185.190.118:31943/v3/OS-FEDERATION/mappings/ACME
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
id	是	String	Mapping的ID。
rules	是	String	将远端用户映射为本地用户的规则列表。
links	是	Object	Mapping的资源链接，包含self。

响应样例

```
{
    "mapping": {
        "id": "ACME",
        "links": {
            "self": "https://example.com/v3/OS-FEDERATION/mappings/ACME"
        },
        "rules": [
            {
                "local": [
                    {
                        "user": {
                            "name": "{0}"
                        }
                    },
                    {
                        "group": {
                            "name": "0cd5e9"
```

```
                        }
                    }
                ],
                "remote": [
                    {
                        "type": "UserName"
                    },
                    {
                        "type": "orgPersonType",
                        "any_one_of": [
                            "Contractor",
                            "SubContractor"
                        ]
                    }
                ]
            }
        ]
    }
}
```

状态码

状态码	说明
200	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
409 Conflict	资源冲突。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.24 删除 Mapping

功能介绍

该接口用于删除Mapping信息。

URI

- URI格式

DELETE /v3/OS-FEDERATION/mappings/{id}

- 参数说明

参数	是否为必选	类型	说明
id	是	String	Mapping的ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	删除已认证用户名下的Mapping时所需的Security Administrator权限token。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X DELETE https://10.185.190.118:31943/v3/OS-FEDERATION/mappings/ACME
```

状态码

状态码	说明
204	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.25 注册 Protocol

功能介绍

该接口用于注册一个Protocol。

URI

- URI格式

PUT /v3/OS-FEDERATION/identity_providers/{idp_id}/protocols/{protocol_id}

- 参数说明

参数	是否为必选	类型	说明
idp_id	是	String	Identity Provider的ID。
protocol _id	是	String	Protocol的ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	为已认证用户创建Protocol时所需的Security Administrator权限token。

- Request Body参数说明

参数	是否为必选	类型	说明
mapping_id	是	String	Mapping的ID。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X PUT -d'{"protocol":{"mapping_id":"ACME"}}' https://10.185.190.118:31943/v3/OS-FEDERATION/identity_providers/ACME/protocols/saml
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
id	是	String	Protocol的ID。
mapping_id	是	String	Mapping的ID。
links	是	Object	Protocol的资源链接，包含identity_provider，self。

响应样例

```
{
    "protocol": {
        "id": "saml",
        "links": {
            "identity_provider": "https://example.com/v3/OS-FEDERATION/identity_providers/ACME",
            "self": "https://example.com/v3/OS-FEDERATION/identity_providers/ACME/protocols/saml"
        },
        "mapping_id": "ACME"
    }
}
```

状态码

状态码	说明
201	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.26 查询 Protocol 列表

功能介绍

该接口用于查询Protocol列表信息。

URI

- URI格式

GET /v3/OS-FEDERATION/identity_providers/{idp_id}/protocols

参数	是否为必选	类型	说明
idp_id	是	String	Identity Provider的ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	查询Protocol列表时所属租户的token。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X GET https://10.185.190.118:31943/v3/OS-FEDERATION/identity_providers/ACME/protocols/
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
protocols	是	List of objects	Protocol的列表。
links	是	Object	Protocol的资源链接，包含next，previous，self。

响应样例

```
{
    "links": {
        "next": null,
        "previous": null,
        "self": "https://example.com/v3/OS-FEDERATION/identity_providers/ACME/protocols"
    },
    "protocols": [
        {
            "id": "saml",
            "links": {
                "identity_provider": "https://example.com/v3/OS-FEDERATION/identity_providers/ACME",
                "self": "https://example.com/v3/OS-FEDERATION/identity_providers/ACME/protocols/saml"
            },
```

```
            "mapping_id": "ACME"
        }
    ]
}
```

状态码

状态码	说明
200	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.27 查询 Protocol

功能介绍

该接口用于查询Protocol信息。

URI

- URI格式

GET /v3/OS-FEDERATION/identity_providers/{idp_id}/protocols/{protocol_id}

- 参数说明

参数	是否为必选	类型	说明
idp_id	是	String	Identity Provider的ID。
protocol _id	是	String	Protocol的ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	查询Protocol时所属租户的token。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X GET https://10.185.190.118:31943/v3/OS-FEDERATION/identity_providers/ACME/protocols/saml
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
id	是	String	Protocol的ID。
mapping_id	是	String	Mapping的ID。
links	是	Object	Protocol的资源链接，包含identity_provider，self。

响应样例

```
{
    "protocol": {
        "id": "saml",
        "links": {
            "identity_provider": "https://example.com/v3/OS-FEDERATION/identity_providers/ACME",
            "self": "https://example.com/v3/OS-FEDERATION/identity_providers/ACME/protocols/saml"
        },
        "mapping_id": "ACME"
    }
}
```

状态码

状态码	说明
200	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。

状态码	说明
405 Method Not Allowed	不允许的方法。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.28 更新 Protocol

功能介绍

该接口用于更新Protocol信息。

URI

- URI格式

PATCH /v3/OS-FEDERATION/identity_providers/{idp_id}/protocols/{protocol_id}

- 参数说明

参数	是否为必选	类型	说明
idp_id	是	String	Identity Provider的ID。
protocol _id	是	String	Protocol的ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	更新已认证用户的Protocol时所需的Security Administrator权限token。

- Request Body参数说明

参数	是否为必选	类型	说明
mapping_id	否	String	Mapping的ID。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X PATCH -d'{"protocol":{"mapping_id":"ACME"}}' https://10.185.190.118:31943/v3/OS-FEDERATION/identity_providers/ACME/protocols/saml
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
id	是	String	Protocol的ID。
mapping_id	是	String	Mapping的ID。
links	是	Object	Protocol的资源链接，包含identity_provider，self。

响应样例

```
{
    "protocol": {
        "id": "saml",
        "links": {
            "identity_provider": "https://example.com/v3/OS-FEDERATION/identity_providers/ACME",
            "self": "https://example.com/v3/OS-FEDERATION/identity_providers/ACME/protocols/saml"
        },
        "mapping_id": "ACME"
    }
}
```

状态码

状态码	说明
200	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
409 Conflict	资源冲突。
413 Request Entity Too Large	请求体过大。

状态码	说明
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.29 删除 Protocol

功能介绍

该接口用于删除Protocol信息。

URI

- URI格式

DELETE /v3/OS-FEDERATION/identity_providers/{idp_id}/protocols/{protocol_id}

- 参数说明

参数	是否为必选	类型	说明
idp_id	是	String	Identity Provider的ID。
protocol _id	是	String	Protocol的ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	删除已认证用户名下的Protocol时所需的Security Administrator权限token。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X DELETE https://10.185.190.118:31943/v3/OS-FEDERATION/identity_providers/ACME/protocols/saml
```

状态码

状态码	说明
204	请求成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
404 Not Found	找不到资源。
405 Method Not Allowed	不允许的方法。
413 Request Entity Too Large	请求体过大。
500 Internal Service Exception	内部服务错误。
503 Service Unavailable	服务不可用。

3.30 导入 Metadata 文件

功能介绍

租户使用联邦认证功能时，需要先将Metadata文件导入IAM中，该接口用于导入租户的Metadata文件。

URI

- URI格式

POST /v3-ext/OS-FEDERATION/identity_providers/{idp_id}/protocols/{protocol_id}/metadata

- 参数说明

参数	是否为必选	类型	说明
idp_id	是	String	已经注册的Identity Provider的ID。
protocol _id	是	String	已经注册的Protocol的ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	更新用户的Metadata文件时所需的Security Administrator权限token。

- Request Body参数说明

参数	是否为必选	类型	说明
xaccount_type	是	String	该字段为标识租户来源字段，默认为空。
metadata	是	String	该字段为用户IDP服务器的Metadata文件的内容。
domain_id	是	String	用户所在domain的ID。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X POST https://10.185.190.118:31943/v3-ext/OS-FEDERATION/identity_providers/ACME/protocols/saml/metadata
```

```
{"xaccount_type":"",
"domain_id":"ed7a77d365304f458f7d0a7909c6d889",
"metadata":"$metadataContent"
}
```

响应

响应样例

```
{ "message": "Import metadata successful"}
```

状态码

状态码	说明
201	导入成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
500 Internal Service Exception	内部服务错误。

3.31 查询 Metadata 文件

功能介绍

该接口用于查询身份提供商导入到IAM中的Metadata文件内容，

URI

- URI格式

GET /v3-ext/OS-FEDERATION/identity_providers/{idp_id}/protocols/{protocol_id}/metadata

- 参数说明

参数	是否为必选	类型	说明
idp_id	是	String	Identity Provider的ID。
protocol _id	是	String	Protocol的ID。

请求

- Request Header参数说明

参数	是否为必选	类型	说明
Content-Type	是	String	该字段内容填为“application/json;charset=utf8”。
X-Auth-Token	是	String	查询用户的Metadata文件内容时所需的Security Administrator权限token。

- 请求样例

```
curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -H "X-Auth-Token:$token" -X GET https://10.185.190.118:31943/v3-ext/OS-FEDERATION/identity_providers/ACME/protocols/saml/metadata
```

响应

- Response Body参数说明

参数	是否为必选	类型	说明
id	是	String	Metadata的ID。
idp_id	是	String	Identity Provider的ID。

参数	是否为必选	类型	说明
entity_id	是	String	Metadata文件中的的entityID字段。
protocol_id	是	String	Protocol的ID。
domain_id	是	String	用户所在domain的ID。
xaccount_type	是	String	表示租户来源的字段，默认为空。
update_time	是	String	导入或更新Metadata文件的时间。
data	是	String	Metadata文件的内容。

响应样例

```
{
"id": "40c174f35ff94e31b8257ad4991bce8b",
"idp_id": "ACME",
"entity_id": "https://idp.test.com/idp/shibboleth",
"protocol_id": "saml",
"domain_id": "ed7a77d365304f458f7d0a7909c6d889",
"xaccount_type": "",
"update_time": "2016-10-26T09:26:23.000000",
"data": "$data"}
```

状态码

状态码	说明
200	获取成功。
400 Bad Request	请求错误。
401 Unauthorized	认证失败。
403 Forbidden	鉴权失败。
500 Internal Service Exception	内部服务错误。

3.32 相关参数信息获取

3.32.1 获取用户名称、项目名称、项目 ID

登录管理控制台，单击右上方登录的账号，选择“我的认证”。进入如下图所示界面，可以查看用户名称、项目名称、项目ID。

A 文档修订记录

表 A-1 文档修订记录

发布时间	修改记录
2016-12-30	第六次正式发布。 本次变更说明如下： ● GET /v3/projects接口的响应增加page字段（查询页面的数据）。 ● GET /v3/projects接口的响应增加per_page字段（页面数据个数）。
2016-10-29	第五次正式发布。 本次变更说明如下： ● 增加**3.15 注册Identity Provider**章节。 ● 增加**3.16 查询Identity Provider列表**章节。 ● 增加**3.17 查询Identity Provider**章节。 ● 增加**3.18 更新Identity Provider**章节。 ● 增加**3.19 删除Identity Provider**章节。 ● 增加**3.20 注册Mapping**章节。 ● 增加**3.21 查询Mapping列表**章节。 ● 增加**3.22 查询Mapping**章节。 ● 增加**3.23 更新Mapping**章节。 ● 增加**3.24 删除Mapping**章节。 ● 增加**3.25 注册Protocol**章节。 ● 增加**3.26 查询Protocol列表**章节。 ● 增加**3.27 查询Protocol**章节。 ● 增加**3.28 更新Protocol**章节。 ● 增加**3.29 删除Protocol**章节。 ● 增加**3.30 导入Metadata文件**章节。 ● 增加**3.31 查询Metadata文件**章节。

发布时间	修改记录
2016-09-30	第四次正式发布。 本次变更说明如下： ● 增加**3.7 查询用户详情**章节。 ● 增加**3.8 创建用户**章节。 ● 增加**3.9 修改用户信息**章节。 ● 增加**3.10 删除用户**章节。 ● 增加**3.11 查询用户列表**章节。 ● 增加**3.12 修改密码**章节。 ● 增加**3.13 查询用户所属用户组**章节。 ● 增加**3.14 查询用户project列表**章节。
2016-08-25	第三次正式发布。 本次变更说明如下： 增加token中各字段的含义。
2016-06-30	第二次正式发布。 本次变更说明如下： ● GET /v3/services接口的响应增加links字段（服务的链接）。 ● GET /v3/endpoints接口的响应增加links字段（终端地址的链接）。
2016-06-17	第一次正式发布。

表 A-2 文档修订记录

发布时间	修改记录
2016-12-30	第一次正式发布。

表 A-3 文档修订记录

发布时间	修改记录
2016-12-30	第一次正式发布。