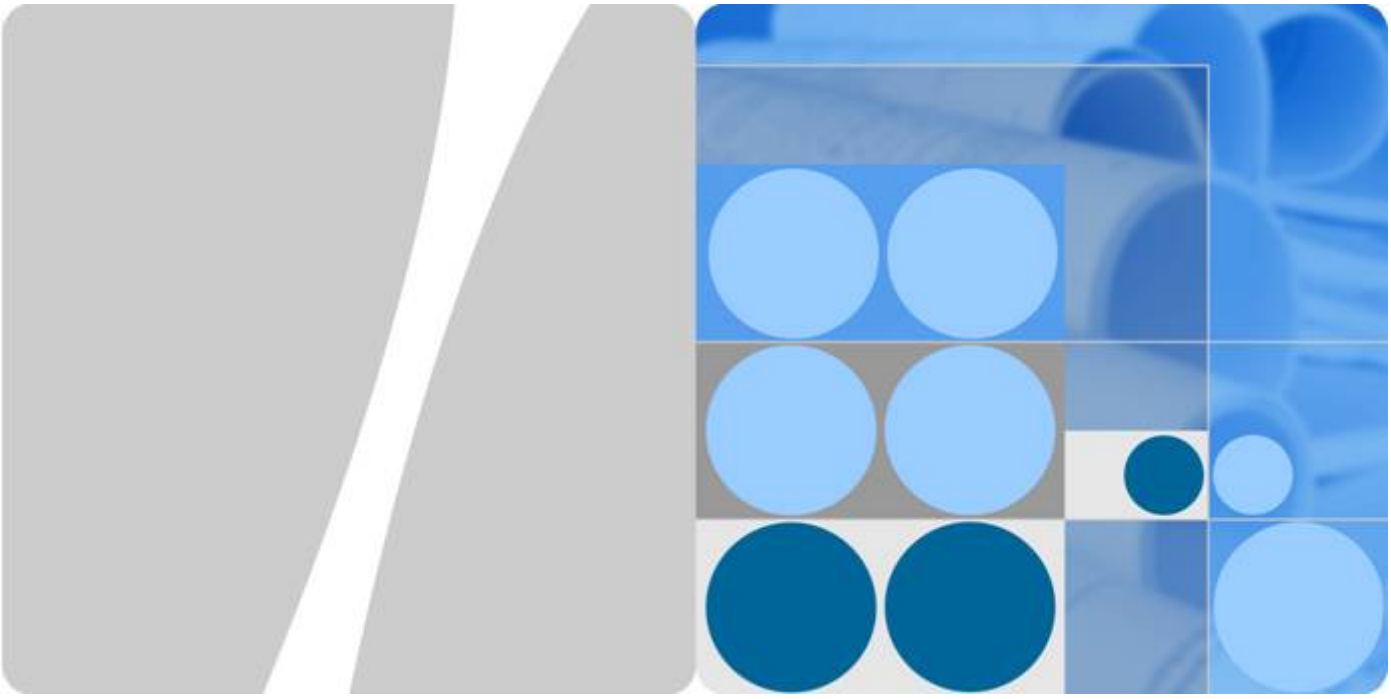


机器学习服务

接口参考

文档版本　01

发布日期　2017-12-04

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://e.huawei.com

目 录

1 接口调用方法

第三方应用对机器学习服务（Machine Learning Service）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

MLS API符合RESTful 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（Uniform Resource Locator）来获取资源。

URL的一般格式为：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
Endpoint	Web服务入口点的URL，从**地区和终端节点** 获取。
uri	资源路径，也即API访问路径。从具体接口的URI模块获取，例如“v3/auth/tokens”。

1.2 请求方法

在HTTP协议中，请求可以使用多种请求方法例如GET、 PUT、 POST、 DELETE、 PATCH，用于指明以何种方式来访问指定的资源，目前提供的REST接口支持的请求方法如**表1-2**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

应用场景

当您使用Token认证方式完成认证鉴权时，需要获取用户Token并在调用接口时增加“X-Auth-Token”到业务接口请求消息头中。

本节介绍如何调用接口完成Token认证。

调用接口步骤

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**。

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《统一身份认证服务API参考》。

```
{
  "auth": {
    "identity": {
      "methods": [
        "password"
      ],
      "password": {
        "user": {
          "name": "username",
          "password": "password",
          "domain": {
            "name": "domainname"
```

```
            }
          }
        }
      },
      "scope": {
        "project": {
          "name": "regioncode"

        }
      }
    }
}
```

说明

请求体中的regioncode，请参见**地区和终端节点**的区域名称。

步骤2 获取Token，请参考《统一身份认证服务API参考》的“获取用户Token”章节。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**步骤2**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(Access Key ID)：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(Secret Access Key)：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

1. 注册并登录管理控制台。
2. 单击用户名，在下拉列表中单击“我的认证”。
3. 单击“管理访问密钥”。
4. 单击“新增访问密钥”，进入“新增访问密钥”页面。
5. 输入当前用户的登录密码。
6. 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。

7. 单击“确定”，在弹出的对话框中，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。

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**。
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

1. 创建用于签名的请求com.cloud.sdk.DefaultRequest(JAVA)。
2. 设置DefaultRequest的目标API URL、HTTPS方法、内容等信息。
3. 对DefaultRequest进行签名：
   a. 调用SignerFactory.getSigner(String serviceName, String regionName)获取一个签名工具实现的实例。
   b. 调用Signer.sign(Request<?> request, Credentials credentials)对**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));
      ```
4. 把上步中签名产生的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“**示例代码**”中的AccessServiceImpl.java。

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。

AccessServiceImpl：实现access方法，具体与API网关通信的代码都在access方法中。

Demo：运行入口，模拟用户进行GET/POST/PUT/DELETE请求。

下面示例代码中的region和serviceName（英文服务名缩写），请参考**地区和终端节点** 。

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 获取项目编号

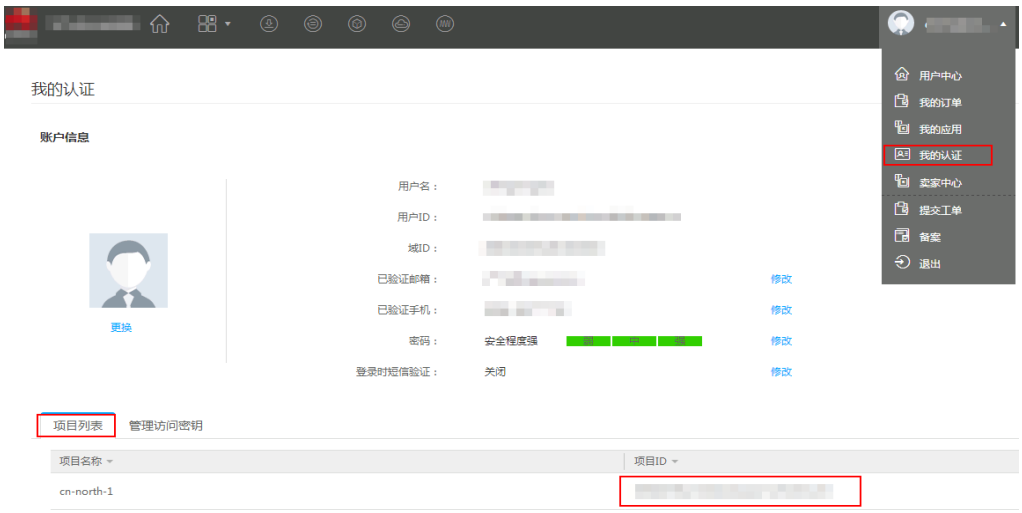
在调用接口的时候，部分URL中需要填入项目编号(project_id或者tenant_id，本文中project_id和tenant_id含义一样)，所以需要先在管理控制台上获取到项目编号。项目编号获取步骤如下：

步骤1 注册并登录管理控制台。

步骤2 单击用户名，在下拉列表中单击“我的认证”。

在“我的认证”页面的项目列表中查看项目ID。

图 1-1 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

名称	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]。端口缺省时使用默认的端口，https的默认端口为443。	否 使用AK/SK认证时必选。	code.test.com or code.test.com:443

名称	描述	是否必选	示例
Content-Type	发送的实体的MIME类型。	是	application/json
Content-Length	请求body长度，单位为Byte。	POST/PUT请求必填。 GET不能包含。	3495
X-Project-Id	project id，用于不同project取token。 如果是DeC的请求则必须传入project id。	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

名称	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	响应消息体的MIME类型。

3 实例管理接口信息

3.1 创建实例

功能介绍

该接口用于创建机器学习服务（Machine Learning Service, MLS）实例。

URI

- URI 格式

```
POST /v1.0/{project_id}/instances
```

- 参数说明

参数	是否为必选	类型	说明
project_id	是	String	项目ID。

请求

- 请求样例

```
POST https://{endpoint}/v1.0/b56e9b448e554ab2aa8833437184fd93/
instances

Request Body:
{
   "instance": {
        "name": "mls-1",
        "version": "1.1.0",
        "flavorRef": "mls.c2.2xlarge.common",
        "network": {
              "vpcId": "85b20d7e-9eb7-4b2a-98f3-3c8843ea3574",
              "subnetId": "374eca02-cfc4-4de7-8ab5-dbebf7d9a720",
              "availableZone": "eu-west-0a",
              "publicIP": {
                    "bindType": "auto_assign",
              }
        },
        "agency": "agency_to_mls",
```

```
  "mrsCluster": {
       "id": "fb8f1161-cf44-47b8-be9c-0f9f73c092a9"
      }
}
```

- 参数说明

参数	是否为必选	类型	说明
instance	是	字典数据结构，详情参见表3-1。	实例定义。

表 3-1 instance 字段说明

参数	是否为必选	类型	说明
name	是	String	实例名称。用于表示实例的名称，同一租户下，同类型的实例名唯一。 取值范围： 4位到64位之间，必须以字母开头，不区分大小写，可以包含字母、数字、中划线或者下划线，不能包含其他的特殊字符。
version	是	String	实例版本。
network	是	字典数据结构，详情参见表3-2。	实例网络信息。
agency	否	String	委托名称，仅在实例绑定EIP时才需要该参数，用于授权MLS服务获取租户token进行EIP绑定实例的操作。 **说明** 委托需要租户在“统一身份认证服务”中提前创建。
flavorRef	是	String	实例规格，如： mls.c2.2xlarge.common
mrsCluster	是	字典数据结构，详情参见表3-4。	实例所关联的MRS集群信息。

表 3-2 network 字段说明

参数	是否为必选	类型	说明
vpcId	是	String	实例所在虚拟私有云ID。
subnetId	是	String	实例所在子网ID。
securityGroupId	是	String	实例所在安全组ID。
availableZone	是	String	可用区域。
publicIP	是	字典数据结构，详情参见表3-3。	公有IP信息。

表 3-3 publicIP 字段说明

参数	是否为必选	类型	说明
bindType	是	String	绑定类型，取值范围： ● auto_assign ● not_use

表 3-4 mrsCluster 字段说明

参数	是否为必选	类型	说明
id	是	String	MRS集群ID。
userName	否	String	MRS集群用户名。 该参数仅在MRS集群为安全模式时才需提供。
userPassword	否	String	MRS集群用户密码，与“username”一起提供配合使用。

响应

- 响应样例

```
{
"instance": {
        "id": "7d85f602-a948-4a30-afd4-e84f47471c15"
```

```
    }
}
```

- 参数说明

参数	是否为必选	类型	说明
instance	是	JSON	新创建的实例信息。
id	是	String	实例ID。

返回值

- 正常

  202

- 异常

返回值	说明
400 Bad Request	请求错误。
401 Unauthorized	鉴权失败。
403 Forbidden	没有操作权限。
404 Not Found	找不到资源。
500 Internal Server Error	服务内部错误。
503 Service Unavailable	服务不可用。

3.2 删除实例

功能介绍

该接口用于删除实例。删除实例将释放实例相关的所有资源，包括租户存储在实例上的数据。

URI

- URI 格式

```
DELETE /v1.0/{project_id}/instances/{instance_id}
```

- 参数说明

参数	是否为必选	类型	说明
project_id	是	String	项目ID。
instance_id	是	String	实例ID。

请求

- 请求样例

```
DELETE https://{endpoint}/v1.0/b56e9b448e554ab2aa8833437184fd93/
instances/7d85f602-a948-4a30-afd4-e84f47471c15

Request Body:
{
}
```

- 参数说明

  无

响应

- 响应样例

```
STATUS CODE 202
```

- 参数说明

  无

返回值

- 正常

  202

- 异常

返回值	说明
400 Bad Request	请求错误。
401 Unauthorized	鉴权失败。
403 Forbidden	没有操作权限。
404 Not Found	找不到资源。
500 Internal Server Error	服务内部错误。
503 Service Unavailable	服务不可用。

3.3 查询实例

功能介绍

该接口用于查询所有已创建的实例列表。

URI

- URI 格式

```
GET /v1.0/{project_id}/instances
```

- 参数说明

参数	是否为必选	类型	说明
project_id	是	String	项目ID。

请求

- 请求样例

```
GET https://{endpoint}/v1.0/b56e9b448e554ab2aa8833437184fd93/
instances
```

- 参数说明

  无

响应

- 响应样例

```
{
    "instances": [
        {
            "id": "7d85f602-a948-4a30-afd4-e84f47471c15",
             "name": "mls-1",
            "version": "1.0.0",
            "status": "Active",
            "currentTask": "Restarting",
            "created": "2017-08-10T14:26:14Z",
            "updated": "2017-08-10T14:26:14Z"
        }
    ]
}
```

- 参数说明

参数	是否为必选	类型	说明
instances	是	字典数据结构。详情参见表3-5。	实例基本信息列表。

表 3-5 instances 字段说明

参数	是否为必选	类型	说明
id	是	String	实例ID。
name	是	String	实例名称。
version	是	String	实例版本。

参数	是否为必选	类型	说明
status	是	String	实例状态，取值范围： ● CREATING ● AVAILABLE ● FAILED ● CREATION FAILED ● FROZEN
currentTask	否	String	实例任务状态，取值范围： ● UNFREEZING ● FREEZING ● RESTORING ● SNAPSHOTTING ● GROWING ● REBOOTING ● REBOOT_FAILURE ● RESIZE_FAILURE
created	是	String	实例创建时间，格式为“yyyy-mm-dd Thh:mm:ssZ”。 其中，T指某个时间的开始；Z指时区偏移量，例如北京时间偏移显示为+0800。
updated	是	String	实例更新时间，格式与“created”相同。

返回值

- 正常

  200
- 异常

返回值	说明
400 Bad Request	请求错误。

返回值	说明
401 Unauthorized	鉴权失败。
403 Forbidden	没有操作权限。
404 Not Found	找不到资源。
500 Internal Server Error	服务内部错误。
503 Service Unavailable	服务不可用。

3.4 查询实例详情

功能介绍

该接口用于查询某个实例的详细信息。

URI

- URI 格式
  ```
  GET /v1.0/{project_id}/instances/{instance_id}
  ```
- 参数说明

参数	是否为必选	类型	说明
project_id	是	String	项目ID。
instance_id	是	String	实例ID。

请求

- 请求样例
  ```
  GET https://{endpoint}/v1.0/b56e9b448e554ab2aa8833437184fd93/
  instances/ 7d85f602-a948-4a30-afd4-e84f47471c15
  ```
- 参数说明

  无

响应

- 响应样例
  ```
  {
      "instance": {
              "id": "7d85f602-a948-4a30-afd4-e84f47471c15",
               "name": "mls-1",
              "version": "1.0.0",
              "status": "Active",
              "currentTask": "Restarting",
              "flavorRef": "mls.c2.2xlarge.common",
              "network": {
                  "vpcId": "85b20d7e-9eb7-4b2a-98f3-3c8843ea3574",
                  "subnetId": "374eca02-cfc4-4de7-8ab5-dbebf7d9a720",
                  "securityGroupId": "dc3ec145-9029-4b39-b5a3-
  ```

```
ace5a01f772b",
                "availableZone": "eu-west-0a",
                "publicIP": {
                    "bindType": "auto_assign",
                    "eipId": "85b20d7e-9eb7-4b2a-98f3-3c8843ea3574"
                }
            },
            "mrsCluster": {
                "id": "fb8f1161-cf44-47b8-be9c-0f9f73c092a9"
            },
            "created": "2017-08-10T14:26:14Z",
            "updated": "2017-08-10T14:26:14Z",
"innerEndPoint": "https://192.168.0.230:22360/mls",
"publicEndPoint": "https://10.154.77.74:22360/mls"
        }
}
```

- 参数说明

参数	是否为必选	类型	说明
instance	是	字典数据结构，详情参见表3-6。	实例详情。

表 3-6 instance 字段说明

参数	是否为必选	类型	说明
id	是	String	实例ID。
name	是	String	实例名称。
version	是	String	实例版本。
flavorRef	是	String	实例规格。
status	是	String	实例状态，取值范围： ● CREATING ● AVAILABLE ● FAILED ● CREATION FAILED ● FROZEN

参数	是否为必选	类型	说明
currentTask	否	String	实例任务状态，取值范围： ● UNFREEZING ● FREEZING ● RESTORING ● SNAPSHOTTING ● GROWING ● REBOOTING ● REBOOT_FAILURE ● RESIZE_FAILURE
network	是	字典数据结构，详情参见表3-7。	实例的网络信息。
mrsCluster	是	字典数据结构，详情参见表3-9。	实例所关联的MRS集群信息。
created	是	String	实例创建时间，格式为“yyyy-mm-dd Thh:mm:ssZ”。 其中，T指某个时间的开始；Z指时区偏移量，例如北京时间偏移显示为+0800。
updated	是	String	实例更新时间，格式与“created”相同。
innerEndPoint	是	String	访问实例的URL地址，该地址只有与实例处在相同VPC和子网的机器能够访问。
publicEndPoint	否	String	访问实例的URL地址，该地址允许从因特网访问。 该地址只有实例绑定EIP时才存在。

表 3-7 network 字段说明

参数	是否为必选	类型	说明
vpcId	是	String	实例所在虚拟私有云ID。
subnetId	是	String	实例所在子网ID。

参数	是否为必选	类型	说明
securityGroupId	是	String	实例所在安全组ID
availableZone	是	String	可用区域。
publicIP	是	字典数据结构，详情参见表3-8。	公有IP信息。

表 3-8 publicIP 字段说明

参数	是否为必选	类型	说明
bindType	是	String	绑定类型，取值范围： ● auto_assign ● not_use
eipId	否	String	弹性IP ID，仅bindType=auto_assign时需要设置。

表 3-9 mrsCluster 字段说明

参数	是否为必选	类型	说明
id	是	String	MRS集群ID。

返回值

- 正常

  200
- 异常

返回值	说明
400 Bad Request	请求错误。
401 Unauthorized	鉴权失败。
403 Forbidden	没有操作权限。
404 Not Found	找不到资源。
500 Internal Server Error	服务内部错误。
503 Service Unavailable	服务不可用。

3.5 查询实例规范

功能介绍

该接口用于查询MLS支持的所有实例规格。

URI

- URI 格式
  ```
  GET /v1.0/{project_id}/flavors
  ```
- 参数说明

参数	是否为必选	类型	说明
project_id	是	String	项目ID。

请求

- 请求样例
  ```
  GET https://{endpoint}/v1.0/b56e9b448e554ab2aa8833437184fd93/flavors
  ```
- 参数说明

  无

响应

- 响应样例
  ```
  STATUS CODE 200

  {
      "flavors": [
          {
              "flavorRef": "mls.c2.2xlarge.common"
          }
      ]
  }
  ```
- 参数说明

参数	是否为必选	类型	说明
flavors	是	列表数据结构，详情参见表3-10。	规格信息列表。

表 3-10 flavors 字段说明

参数	是否为必选	类型	说明
flavorRef	是	String	规格名称。

返回值

- 正常

  200
- 异常

返回值	说明
400 Bad Request	请求错误。
401 Unauthorized	鉴权失败。
403 Forbidden	没有操作权限。
404 Not Found	找不到资源。
500 Internal Server Error	服务内部错误。
503 Service Unavailable	服务不可用。

3.6 查询版本信息

功能介绍

该接口用于查询MLS可用版本列表。

URI

- URI 格式

  ```
  GET /v1.0/{project_id}/versions
  ```
- 参数说明

参数	是否为必选	类型	说明
project_id	是	String	项目ID。

请求

- 请求样例

  ```
  GET https://{endpoint}/v1.0/b56e9b448e554ab2aa8833437184fd93/versions
  ```
- 参数说明

  无

响应

- 响应样例

  ```
  STATUS CODE 200

  {
      "versions": [
          {
  ```

```
            "version": "1.1.0"
        }
    ]
}
```

- 参数说明

参数	是否为必选	类型	说明
versions	是	列表数据结构，详情参见表3-11。	版本信息列表。

表 3-11 versions 字段说明

参数	是否为必选	类型	说明
version	是	String	版本编码。
description	否	String	版本描述。

返回值

- 正常

  200

- 异常

返回值	说明
400 Bad Request	请求错误。
401 Unauthorized	鉴权失败。
403 Forbidden	没有操作权限。
404 Not Found	找不到资源。
500 Internal Server Error	服务内部错误。
503 Service Unavailable	服务不可用。

A 附录

A.1 异常响应

MLS API提供统一的异常响应，它由错误码和异常信息组成。

响应

- 响应样例

```
STATUS CODE 500
{
    "errCode": "MLS.0016",
    "externalMessage": "cannot find the resource"
}
```

- 参数说明

参数	类型	说明
errCode	String	错误码。
externalMessage	String	异常信息。

A.2 错误码说明

错误码	响应码	错误信息
MLS.0001	400	参数错误！
MLS.0005	500	服务器错误。
MLS.0006	400	请求为空，请输入请求参数。
MLS.0011	409	该实例正在进行其它操作或该实例故障，无法执行该操作，请稍后重试。
MLS.0015	403	访问资源不存在，或者无访问权限。

错误码	响应码	错误信息
MLS.0020	403	Restricted account.
MLS.0021	403	Frozen account.
MLS.0022	404	实例不存在或已删除。
MLS.0032	403	You are not authorized to perform the operation. Check the account permission on the IAM.
MLS.1102	400	实例名称已存在！
MLS.1112	400	实例数量达到配额。
MLS.2067	400	输入参数不合法！
MLS.5001	400	实例名不合法！
MLS.5006	400	所选规格非法！
MLS.5007	404	所选规格不存在！
MLS.5014	400	Vpc id非法！
MLS.5015	400	子网id非法！
MLS.5021	404	Vpc不存在，或不属于该用户！
MLS.5022	404	子网不存在，或不属于该vpc！
MLS.5026	400	Ha available zone is illegal!
MLS.5027	400	Ha invalid available zone!
MLS.5045	400	当前租户不允许操作！
MLS.5048	400	扩展参数不合法！
MLS.5049	400	集群名非法。
MLS.5050	409	集群名已经存在。
MLS.5052	400	可用区取值非法！
MLS.5055	400	所选规格ID取值非法！
MLS.5060	400	Integer 类型长度违规。
MLS.5063	400	Specific 字段无效。
MLS.5070	400	规格信息与xml配置不匹配。
MLS.5074	400	子网不属于vpc。
MLS.5097	404	HA Region 或 Available Zone 不存在！
MLS.5098	400	Invalid EIP binding type.
MLS.9999	500	请求处理失败。

错误码	响应码	错误信息
MLS.10001	400	Invalid MRS cluster id.
MLS.10002	400	MRS cluster version is not supported.
MLS.10003	400	MRS cluster is unavailable.
MLS.10004	400	Invalid MRS user name.
MLS.10005	400	Invalid MRS user password.
MLS.10006	400	Fail to get MRS cluster info.
MLS.10007	400	Invalid MLS version.

B 文档修订记录

发布日期	修订记录
2017-12-4	第一次正式发布。