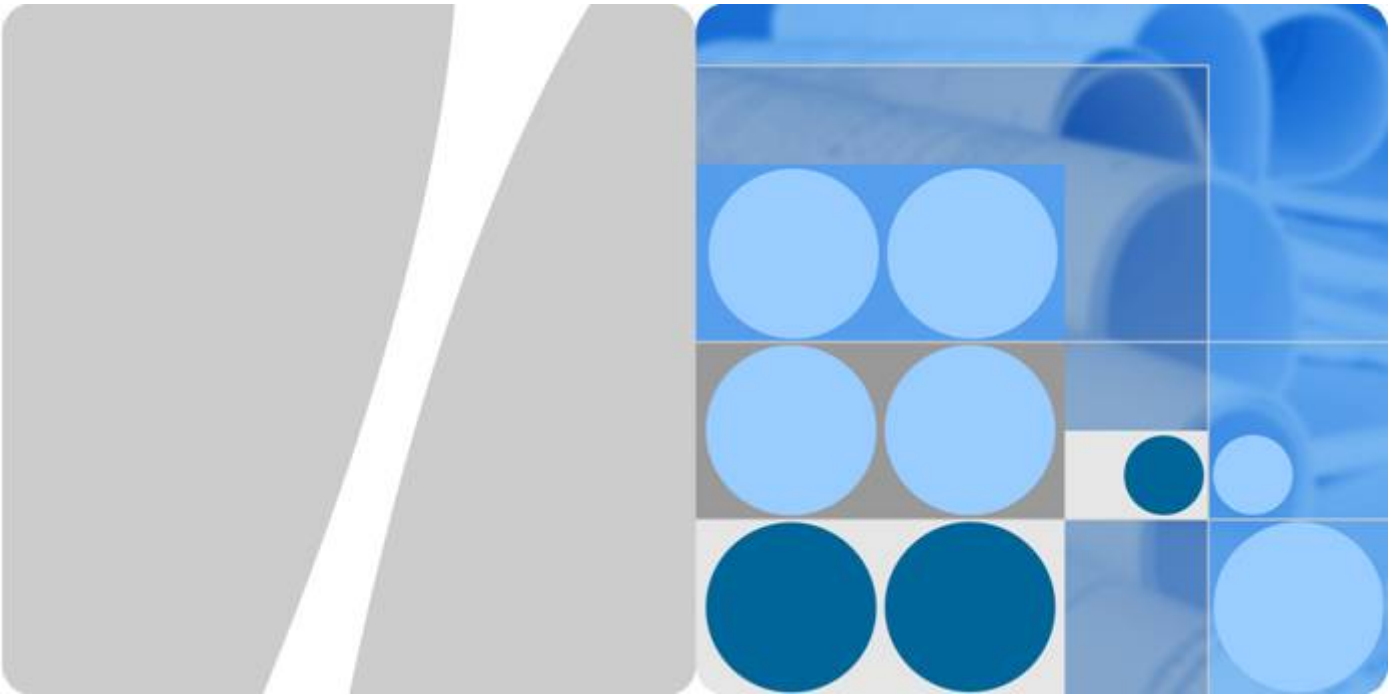


Image Recognition

SDK Reference

Issue 10

Date 2020-03-30

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

Trademarks and Permissions


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

Notice

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

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

Huawei Technologies Co., Ltd.

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

Website: https://www.huawei.com

Email: support@huawei.com

Contents

1 Introduction to Image Recognition SDK

Overview of Image Recognition

Image Recognition is a technology that uses computers to process, analyze, and understand images to identify objects in different modes. It can also improve the image quality with the support of enhancement and reconstruction technologies.

Image Recognition provides services through open application programming interfaces (APIs). You can obtain the inference results by accessing and calling APIs in real time. It helps you collect key data automatically and build an intelligent service system, thereby improving service efficiency.

SDK Overview

Image Recognition Software Development Kit (Image SDK for short) encapsulates the RESTful APIs provided by Image Recognition to simplify application development. You can directly call API functions provided by Image SDK to use Image Recognition.

Mappings Between Services and APIs

Table 1-1 lists the mappings between the Image Recognition sub-services and APIs.

Table 1-1 Mappings between services and APIs

Service	API
Image Tagging	POST /v1.0/image/tagging
Celebrity Recognition	POST /v1.0/image/celebrity-recognition
Recapture Detection	POST /v1.0/image/recapture-detect
Dark Enhance	POST /v1.0/vision/dark-enhance
Defog	POST /v1.0/vision/defog
Super Resolution	POST /v1.0/vision/super-resolution

2 Applying for a Service

For details about how to apply for the service, see **Applying for a Service** in the *Image Recognition API Reference*.

3 Obtaining Authentication Information

Service APIs need to be authenticated. Two authentication methods, token authentication and AK/SK authentication, are available. You are advised to use AK/SK authentication.

Step 1 Register with and log in to the management console.

Step 2 Hover the cursor on the username and select **My Credential** from the drop-down list.

Step 3 Click the **Access Keys** tab and then click **Add Access Key**.

Step 4 Enter the verification code sent to your mail or mobile phone.

Step 5 Click **OK** to download the AK/SK of the authentication account. The AK/SK data is saved as a local file. Keep the file properly.

----End

4 Obtaining Image SDK

Download the SDK software package and related document of Image Recognition at **HUAWEI CLOUD SDKs**.

Obtain the endpoint of Image Recognition at **Regions and Endpoints**.

5 Preparing the Environment

Before using the Image Recognition SDK, you need to prepare the environment. The Java SDK is used as an example. For details, see **Table 5-1**.

Table 5-1 Development environment

Item	Description
Operating system (OS)	Windows OS. Windows 7 or later is recommended.
JDK installation	Basic configuration for the Java development environment. The 1.8 version is strongly recommended.

6 Using Java SDK

6.1 Preparing a Java Development Environment

Image Java SDK uses Java SE Development Kit 8 (JDK 8) or later. The following uses JDK 8 (Windows x64) running on Windows 7 as an example. If you have downloaded the JDK and configured the environment, skip this section.

Step 1 **Download the JDK file**.

Step 2 After the JDK file is downloaded, install the JDK as prompted. For example, install the JDK to the **C:\Program Files\Java\jdk1.8.0_131** directory on the local PC.

Step 3 Right-click **Computer**, choose **Properties** > **Advanced System Settings** > **Environment Variables**, and perform the following operations to configure Java environment variables:

1. Create system variable **JAVA_HOME** whose value is the JDK installation path.
2. Add **%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin** to **Path**. Separate multiple values with semicolons (;).
3. Create system variable **CLASSPATH** whose value is **%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar**.

Step 4 Open the command line interface (CLI) and run **java -version**. If the information shown in **Figure 6-1** is displayed, the configuration is successful.

Figure 6-1 Java version information

```
C:\>java -version
java version "1.8.0_131"
Java(TM) SE Runtime Environment (build 1.8.0_131-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.131-b11, mixed mode)
```

----End

6.2 Installing Eclipse and Importing SDK Projects

The following uses Eclipse as an example to describe how to import the SDK project. Operations of other IDEs are similar to those of Eclipse.

Step 1 **Download the eclipse version of the corresponding platform.** For example, **eclipse-jee-mars-R-win32-x86_64.zip**.

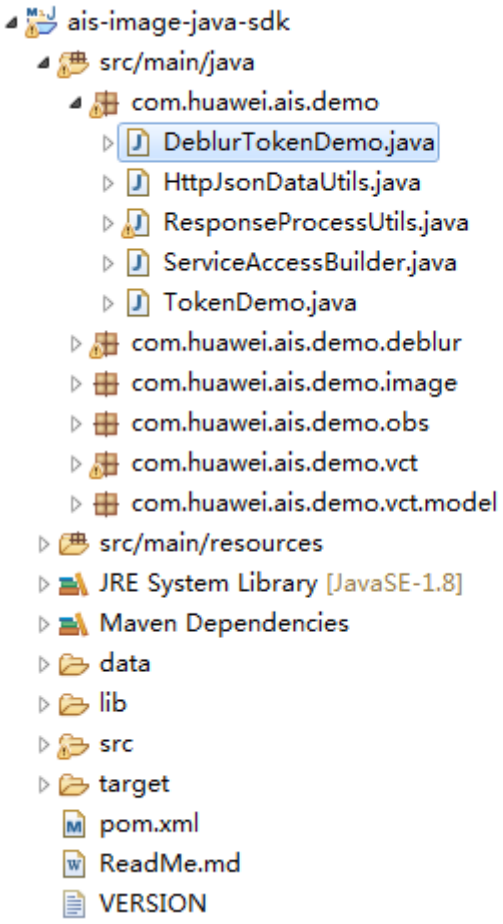
Step 2 Extract and open Eclipse.

Step 3 Configure the correct JRE path in **Windows** > **Preferences** > **Java** > **Installed JREs**.

Step 4 Right-click **Package Explorer** on the left, and click **Import**. Choose **Maven** > **Existing Maven Projects**, and click **Next**. Then click **Browse**, and select the path where **ais-image-java-sdk** resides.

Step 5 Click **Finish** to import the SDK. After the SDK is imported, open the project. **Figure 6-2** shows the project directory.

Figure 6-2 Project directory



When Maven is used for build, the JAR package in the **lib** directory of the SDK package is the local JAR package on which the project depends. Other

dependencies can be configured in the **pom.xml** file and obtained from the central repository or Huawei open-source image site.

----End

6.3 Demo Project of Image Tagging

Two authentication methods, token authentication and AK/SK authentication, are available. This section uses AK/SK authentication as an example.

This demo project corresponds to the **POST /v1.0/image/tagging** URI. Replace the AK/SK information with the actual AK/SK to run the demo.

Step 1 Configure the AK/SK in the **ImageTaggingDemo.java** file. The sample code is as follows:

```
// 1. Configure the basic information for accessing Image Tagging and generate a client connection object.
AisAccess service = ServiceAccessBuilder.builder()
        .ak("######")                    // your ak
        .sk("######")                    // your sk
        .region("cn-north-1")             // Configuration of Image Recognition in the CN North-Beijing1 region
        .connectionTimeout(5000)            // Timeout limit for connecting to the target URL
        .connectionRequestTimeout(1000)     // Timeout limit for obtaining available connections from the
connection pool
        .socketTimeout(20000)             // Timeout limit for obtaining server response data
        .build();
```

Step 2 Select a local image or use the default image of the sample project, and modify the image path (*data/image-tagging-demo-1.jpg*) in the **ImageTaggingDemo.java** file.

```
//
// 2. Construct the parameters required for accessing Image Tagging.
//
String uri = "/v1.0/image/tagging";
byte[] fileData = FileUtils.readFileToByteArray(new File("data/image-tagging-demo-1.jpg"));
String fileBase64Str = Base64.encodeBase64String(fileData);
```

Step 3 Execute the **ImageTaggingDemo.java** file. If **200** is displayed on the console, the program is successfully executed. The **recognization result** is shown in #image_04_0008/d0e652.

Figure 6-3 Execution result

```
"D:\Program Files\Java\jdk1.8.0_191\bin\java.exe" ...
200
{
    "result":{
        "tags":[
            {
                "confidence":"100.0",
                "tag":"Koala",
                "type":"object",
                "i18n_tag":{
                    "en":"Koala"
                }
            },
            {
                "confidence":"100.0",
                "tag":"Marsupials",
                "type":"object",
                "i18n_tag":{
                    "en":"Marsupials"
                }
            },
            {
                "confidence":"70.51",
                "tag":"Veterinarians office",
                "type":"scene",
                "i18n_tag":{
                    "en":"Veterinarians office"
                }
            }
        ]
    }
}
```

----End

6.4 Demo Project of Celebrity Recognition

Two authentication methods, token authentication and AK/SK authentication, are available. This section uses AK/SK authentication as an example.

This demo project corresponds to the **POST /v1.0/image/celebrity-recognition** URI. Replace the AK/SK information with the actual AK/SK to run the demo.

Step 1 Configure the AK/SK in the **CelebrityRecognitionDemo.java** file. The sample code is as follows:

```
// 1. Configure the basic information for accessing Celebrity Recognition and generate a client connection object.
AisAccess service = ServiceAccessBuilder.builder()
        .ak("######")                    // your ak
        .sk("######")                    // your sk
        .region("cn-north-1")             // Configuration of Image Recognition in the CN North-Beijing1 region
        .connectionTimeout(5000)          // Timeout limit for connecting to the target URL
        .connectionRequestTimeout(1000)   // Timeout limit for obtaining available connections from the connection pool
        .socketTimeout(20000)             // Timeout limit for obtaining server response data
        .build();
```

Step 2 Select an image of a political figure or Internet celebrity, and modify the image path (*data/celebrity-recognition.jpg*) in the **CelebrityRecognitionDemo.java** file.

```
//
// 2. Construct the parameters required for accessing Celebrity Recognition.
//
String uri = "/v1.0/image/celebrity-recognition";
/**
```



```
* TODO In this example, the image to be recognized can be placed in the data directory. Replace the name of the example image with the name of the image to be recognized.
*/
byte[] fileData = FileUtils.readFileToByteArray(new File("data/celebrity-recognition.jpg"));
String fileBase64Str = Base64.encodeBase64String(fileData);
```

Step 3 Execute the **CelebrityRecognitionDemo.java** file. If **200** is displayed on the console, the program is successfully executed. The **recognition result** is shown in **Figure 6-4**.

Figure 6-4 Execution result

```
"D:\Program Files\Java\jdk1.8.0_191\bin\java.exe"
200
{
    "result":[
        {
            "confidence":0.9989172992292055,
            "face_detail":{
                "w":365,
                "h":514,
                "x":260,
                "y":198
            },
            "label":"Michael Jackson in Chinese"
        }
    ]
}
```

----End

6.5 Demo Project of Recapture Detection

Two authentication methods, token authentication and AK/SK authentication, are available. This section uses AK/SK authentication as an example.

This demo project corresponds to the **POST /v1.0/image/recapture-detect** URI. Replace the AK/SK information with the actual AK/SK to run the demo.

Step 1 Configure the AK/SK in the **RecaptureDetectionDemo.java** file. The sample code is as follows:

```
// 1. Configure the basic information for accessing Recapture Detection and generate a client connection object.
AisAccess service = ServiceAccessBuilder.builder()
        .ak("######")                    // your ak
        .sk("######")                    // your sk
        .region("cn-north-1")             // Configuration of Image Recognition in the CN North-Beijing1 region
        .connectionTimeout(5000)          // Timeout limit for connecting to the target URL
        .connectionRequestTimeout(1000)   // Timeout limit for obtaining available connections from the connection pool
        .socketTimeout(20000)             // Timeout limit for obtaining server response data
        .build();
```

Step 2 Select a local image or use the default image of the sample project, and modify the image path (*data/recapture-detect-demo-1.jpg*) in the **RecaptureDetectionDemo.java** file.

```
//
// 2. Construct the parameters required for accessing Recapture Detection.
//
String uri = "/v1.0/image/recapture-detect";
```



```
byte[] fileData = FileUtils.readFileToByteArray(new File("data/recapture-detect-demo-1.jpg"));
String fileBase64Str = Base64.encodeBase64String(fileData);
```

Step 3 Execute the **RecaptureDetectionDemo.java** file. If **200** is displayed on the console, the program is successfully executed. The **recognization result** is shown in **Figure 6-5**.

Figure 6-5 Execution result

```
"C:\Program Files\Java\jdk1.8.0_112\bin\java" ...
200
{"result":{"suggestion":"false","category":"recapture","score":"1.0","detail":[{"label":"recapture","confidence":"1.0"}]}}
```

----End

6.6 Demo Project of Dark Enhance

Two authentication methods, token authentication and AK/SK authentication, are available. This section uses AK/SK authentication as an example.

This demo project corresponds to the **POST /v1.0/vision/dark-enhance** URI. Replace the AK/SK information with the actual AK/SK to run the demo.

Step 1 Configure the AK/SK in the **DarkEnhanceDemo.java** file. The sample code is as follows:

```
// 1. Configure the basic information for accessing Dark Enhance and generate a client connection object.
AisAccess service = ServiceAccessBuilder.builder()
        .ak("######")                    // your ak
        .sk("######")                    // your sk
        .region("cn-north-1")             // Configuration of Image Recognition in the CN North-Beijing1 region
        .connectionTimeout(5000)            // Timeout limit for connecting to the target URL
        .connectionRequestTimeout(1000)     // Timeout limit for obtaining available connections from the
connection pool
        .socketTimeout(20000)              // Timeout limit for obtaining server response data
        .build();
```

Step 2 Select a local image or use the default image of the sample project, and modify the original image path (*data/dark-enhance-demo-1.bmp*) and the path for saving the image generated after processing (*data/dark-enhance-demo-1.cooked.bmp*) in the **DarkEnhanceDemo.java** file.

```
//
// 2. Construct the parameters required for accessing Dark Enhance.
//
String uri = "/v1.0/vision/dark-enhance";
byte[] fileData = FileUtils.readFileToByteArray(new File("data/dark-enhance-demo-1.bmp"));
String fileBase64Str = Base64.encodeBase64String(fileData);

JSONObject json = new JSONObject();
json.put("image", fileBase64Str);
json.put("brightness", 0.9);

// 3. Pass the URI and required parameters of Dark Enhance.
// Pass the parameters in JSON objects and call the service using POST.
HttpResponse response = service.post(uri, json.toJSONString());
// 4. Check whether the API call is successful. If 200 is returned, the API call succeeds. Otherwise, it fails.
ResponseProcessUtils.processResponseStatus(response);
// 5. Process the character stream returned by the service and create the image file processed by Dark
Enhance.
ResponseProcessUtils.processResponseWithImage(response, "data/dark-enhance-demo-1.cooked.bmp");
```


Step 3 Execute the **DarkEnhanceDemo.java** file to output the **recognition result** to the console. If **200** is displayed on the console, the program is successfully executed.

Figure 6-6 shows the original image and enhanced image.

Figure 6-6 Comparison between the original image and enhanced image

----End

6.7 Demo Project of Defog

Two authentication methods, token authentication and AK/SK authentication, are available. This section uses AK/SK authentication as an example.

This demo project corresponds to the **POST /v1.0/vision/defog** URI. Replace the AK/SK information with the actual AK/SK to run the demo.

Step 1 Configure the AK/SK in the **DefogDemo.java** file. The sample code is as follows:

```
// 1. Configure the basic information for accessing Defog and generate a client connection object.
AisAccess service = ServiceAccessBuilder.builder()
        .ak("######")                    // your ak
        .sk("######")                    // your sk
        .region("cn-north-1")             // Configuration of Image Recognition in the CN North-Beijing1 region
        .connectionTimeout(5000)          // Timeout limit for connecting to the target URL
        .connectionRequestTimeout(1000)     // Timeout limit for obtaining available connections from the
connection pool
        .socketTimeout(20000)             // Timeout limit for obtaining server response data
        .build();
```

Step 2 Select a local image or use the default image of the sample project, and modify the original image path (*data/defog-demo-1.png*) and the path for saving the image generated after processing (*data/defog-demo-1.cooked.png*) in the **DefogDemo.java** file.

```
//
// 2. Construct the parameters required for accessing Defog.
//
String uri = "/v1.0/vision/defog";
byte[] fileData = FileUtils.readFileToByteArray(new File("data/defog-demo-1.png"));
String fileBase64Str = Base64.encodeBase64String(fileData);
```



```
JSONObject json = new JSONObject();
json.put("image", fileBase64Str);
json.put("gamma", 1.5);
json.put("natural_look", true);
// 3. Pass the URI and required parameters of Defog.
// Pass the parameters in JSON objects and call the service using POST.
HttpResponse response = service.post(uri, json.toJSONString());
// 4. Check whether the API call is successful. If 200 is returned, the API call succeeds. Otherwise, it fails.
ResponseProcessUtils.processResponseStatus(response);
// 5. Process the character stream returned by the service and create the image file processed by Defog.
ResponseProcessUtils.processResponseWithImage(response, "data/defog-demo-1.cooked.png");
```

Step 3 Execute the **DefogDemo.java** file. The **recognition result** is output. If **200** is displayed on the console, the program is successfully executed. **Figure 6-7** shows the original image and defogged image.

Figure 6-7 Comparison between the original image and defogged image

----End

6.8 Demo Project of Super Resolution

Two authentication methods, token authentication and AK/SK authentication, are available. This section uses AK/SK authentication as an example.

This demo project corresponds to the **POST /v1.0/vision/super-resolution** URI. Replace the AK/SK information with the actual AK/SK to run the demo.

Step 1 Configure the AK/SK in the **SuperResolutionDemo** file. The sample code is as follows:

```
// 1. Configure the basic information for accessing Super Resolution and generate a client connection object.
AisAccess service = ServiceAccessBuilder.builder()
        .ak("######")                    // your ak
        .sk("######")                    // your sk
        .region("cn-north-1")             // Configuration of Image Recognition in the CN North-Beijing1 region
        .connectionTimeout(5000)          // Timeout limit for connecting to the target URL
        .connectionRequestTimeout(1000)   // Timeout limit for obtaining available connections from the
connection pool
        .socketTimeout(20000)             // Timeout limit for obtaining server response data
        .build();
```

Step 2 Select a local image or use the default image of the sample project, and modify the original image path (*data/super-resolution-demo-1.png*) and the path for

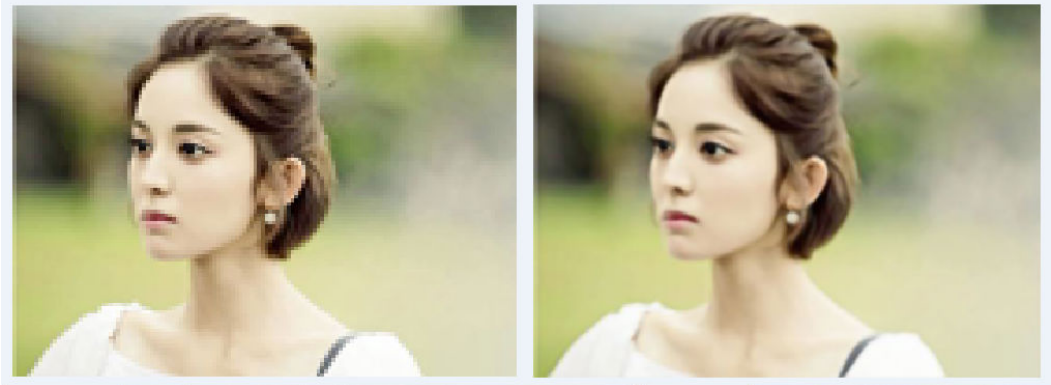
saving the image generated after processing (*data/super-resolution-demo-1.cooked.png*) in the **SuperResolutionDemo.java** file.

```
//
// 2. Construct the parameters required for accessing Super Resolution.
//
String uri = "/v1.0/vision/super-resolution";
byte[] fileData = FileUtils.readFileToByteArray(new File("data/super-resolution-demo-1.png"));
String fileBase64Str = Base64.encodeBase64String(fileData);

JSONObject json = new JSONObject();
json.put("image", fileBase64Str);
json.put("scale", 3);
json.put("model", "ESPCN");

// 3. Pass the URI and required parameters of Super Resolution.
// Pass the parameters in JSON objects and call the service using POST.
HttpResponse response = service.post(uri, json.toJSONString());
// 4. Check whether the API call is successful. If 200 is returned, the API call succeeds. Otherwise, it fails.
ResponseProcessUtils.processResponseStatus(response);
// 5. Process the character stream returned by the service and create the image file processed by Super Resolution.
ResponseProcessUtils.processResponseWithImage(response, "data/super-resolution-demo-1.cooked.png");
```

Step 3 Execute the **SuperResolutionDemo.java** file. The **recognition result** is output. If **200** is displayed on the console, the program is successfully executed.

Figure 6-8 shows the original image and reconstructed image.

Figure 6-8 Comparison between the original image and reconstructed image



----End

6.9 SDK Project Using Token Authentication

Two authentication methods, token authentication and AK/SK authentication, are available. This section uses token authentication as an example.

(You can use Image Tagging after performing the following operations. You only need to change the username and password instead of compiling code.)

Step 1 Open the **TokenDemo.java** file in the **com.huawei.ais.demo** package, and change the values of **username** and **password** in the **main** function to the actual username and password registered with the system. The sample code is as follows:

```
/**
 * Invoke the main entrypoint function.
 */
public static void main(String[] args) throws URISyntaxException, UnsupportedOperationException,
```



```
IOException {
    String username = "zhangshan";   // Enter the username.
    String password = "*******";     // Enter the corresponding password.

    String token = getToken(username, password, projectName);
    System.out.println(token);
    requestImageTaggingBase64(token, "data/image-tagging-demo-1.jpg");
    }
```

Step 2 After modification, run the **TokenDemo.java** file. On the console, you can see the recognition result using token authentication.

----End

7 Change History

Release Date	What's New
2019-05-15	This is the tenth official release. Modified the following content: • **Obtaining Authentication Information** • **Using Java SDK**
2019-01-31	This is the ninth official release. Modified the following content: • Changed the endpoint of Image Recognition to **image.cn-north-1.myhuaweicloud.com**.
2018-12-14	This is the eighth official release.
2018-11-15	This is the seventh official release. Added the following content: • **Introduction to Image Recognition SDK** • **Preparing the Environment**
2018-10-30	This is the sixth official release. Added the following content: • **Demo Project of Celebrity Recognition**
2018-09-19	This is the fifth official release. Modified the following content: • Changed the reference document about how to apply for a service to the *Image Recognition API Reference*.
2018-07-16	This is the fourth official release. Modified the following content: • Modified the project import path in **Installing Eclipse and Importing SDK Projects**.

Release Date	What's New
2018-07-02	This is the third official release. Added the following content: • **Demo Project of Dark Enhance** • **Demo Project of Defog** • **Demo Project of Super Resolution**
2018-04-20	This is the second official release. Modified the following content: • Modified the obtained information in **Applying for a Service**.
2018-01-03	This is the first official release.