DataArts Studio

SDK Reference

Issue 01

Date 2022-09-30

HUAWEI

HUAWEI TECHNOLOGIES CO., LTD.

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 DataArts DataService SDK Reference

1.1 Overview

This document provides guidance for you to call APIs through App authentication.

When the App authentication is used, the SDK is required for access.

1.2 Preparations for Using the SDK

1. Download the SDK and import it to a local development tool. For details, see **Preparation**.
2. Set parameters such as **appKey**, **appSecret**, **method**, and **url**.

Table 1-1 Parameters

Type	Description	Example
Path parameter	The path parameter is part of the URL. Use it to replace the parameter in {} in the URL.	Parameter: param = xxx Original URL: http://*Domain name*/p1/{param}/p2 Actual URL: http://*Domain name*/p1/xxx/p2

Type	Description	Example
Query parameter	The query parameter is a supplementary part of the URL.	Parameter: param = xxx Parameter 2: param2 = xxx2 Example 1: Add the query parameter to a method (use the SDK of each language as an example). Example: request.addQueryStringParam("param"," xxx"); Example 2: Add a question mark (?) and the parameter to the end of the URL. If there are multiple parameters, separate them with ampersands (&). Original URL: http://*Domain name*/p1 Actual URL: http://*Domain name*/p1?param=xxx&param2=xxx2
Header parameter	The header parameter is part of the request header. The parameter name is case insensitive.	Parameter: param = xxx Add a header parameter to a method or add a header parameter when constructing a request (subject to the SDK of each language). Example: request.addHeader("param"," xxx");
Body parameter	The request body parameter is a JSON string in the SDK. This parameter is unavailable in earlier versions.	"{}"

3. Modify the SDK and obtain the signature parameter **Authorization** in the request header after the request is signed. In addition, add the **x-Authorization** parameter with the same value as **Authorization**. For details about how to obtain the **Authorization** parameter and add the **x-Authorization** parameter, see **Preparation**.

1.3 Common Error Codes and Messages for SDK Invocation

Table 1-2 Error codes and messages

Error Code	Error Message	Error Cause	Solution
APIG.0101	The API does not exist or has not been published in the environment	1. The API has not been published. 2. The URL is incorrect.	1. Publish the API. 2. Ensure that the request URL is correct.
APIG.0106	Orchestration error: Invalid header parameter: x-Authorization, required	x-Authorization is not added to the SDK.	See **Step 3 in Preparations for Using the SDK**.
APIG.0106	Orchestration error: Invalid ___ parameter: ___, required	A specified parameter is not transferred.	Transfer the parameter during invocation.
APIG.0201	Backend timeout	API Gateway does not receive a response within 50 seconds after sending a request.	Check the DataArts DataService access log. If the access log contains data (the data is slightly delayed), the data source extraction time is too long. In this case, optimize the data extraction SQL logic. If the access log does not contain data, check whether the DataArts DataService Exclusive cluster is running.

Error Code	Error Message	Error Cause	Solution
APIG.0303	Incorrect app authentication information: app not found	The application does not exist.	Check whether the request key and secret are correct.
APIG.0304	The app is not authorized to access the API	The application does not have the permission to access the current API.	1. Ensure that the application has been authorized to access the API. 2. Check whether the request key and secret are correct.
APIG.0308	The throttling threshold has been reached: policy domain over ratelimit, limit:1000, time:1 day	The number of domain name requests has reached the upper limit, which is 1,000 per day.	1. Suggestion: Bind a domain name to the API group in API Gateway. 2. Workaround: Change another group. Domain names are grouped. Each group has an upper limit.

Error Code	Error Message	Error Cause	Solution
DLM.4018	Api is not exist	The API does not exist.	For APIs released before the version on June 30, 2020: Check whether the value of **x-api-id** is correct. (The value is the ID of the API to be accessed and can be obtained from the API provider.) For APIs released after the version on June 30, 2020: 1. Ensure that the request URL is correct. 2. If the API is just published by DataArts DataService Exclusive, wait for a while. There is a short delay before the API is delivered to the cluster. Other APIs (data synchronization exception): 1. Disable or suspend the API, and then resume or publish the API. 2. Restart the cluster. (Restart nodes one by one to avoid impact on services.)

Error Code	Error Message	Error Cause	Solution
DLM.4094	Call api failed.	The API fails to be called.	1. Ensure that the SQL statement used to call the API is correct and can be executed properly. (For details about the SQL statement, see the access log. The SQL statement is visible only to the API caller.) 2. The CDM agent is abnormal. For details about the error cause, see the returned DLG error message. 3. The API call times out. If the DWS database is used, you are advised to use the custom pagination mode. 4. The API call times out. Optimize the query statement to ensure that it can be executed in the database in a short time.

Error Code	Error Message	Error Cause	Solution
DLM.4211	Token invalid	Token verification fails.	1. Check whether the token is correct. 2. Check whether the tenant to which the token belongs has been authorized or is in the allowlist.
DLM.4312	Missing parameters: ___	A specified parameter is missing.	Transfer the parameter during invocation.
400	App does not have permission to access API.	The application does not have the permission to access the current API.	1. Ensure that the application has been authorized to access the API. 2. Check whether the request key and secret are correct. 3. Ensure that the authorization relationship between the API and application is still valid.

Error Code	Error Message	Error Cause	Solution
401	Authorization not found.	Authorization information is not found.	1. Application authentication: **Step 3 in Preparations for Using the SDK** 2. IAM authentication for an API in DataArts DataService Exclusive published to the gateway: An API using the IAM authentication method cannot directly access the cluster through token authentication.
401	Authorization format incorrect.	The authorization format is incorrect.	You are advised to use the SDK to generate a signature.
401	Signing key not found.	The signature key is not found.	Check whether the request key and secret are correct.
401	Signed header ___ not found.	The signature header is not found.	Ensure that the header parameter used for signature is uploaded during the SDK invocation.
401	Header x-sdk-date not found.	The **x-sdk-date** header is not found.	This parameter is automatically generated when the SDK is signed. If the SDK is invoked in other ways, upload the parameter when invoking the signed SDK.

Error Code	Error Message	Error Cause	Solution
401	Signature expired.	The signature has expired.	1. The signature has a validity period. If the signature has expired, generate a new one. 2. Check whether the local time is the same as the actual time. 3. If the local time is correct, contact related personnel to check whether the time of the cluster nodes is normal.

Error Code	Error Message	Error Cause	Solution
401	Verify authorization failed.	Signature verification fails.	Ensure that all signature parameters have been uploaded and are the same as those used during the signature, including but not limited to **url**, **path**, **header**, **query**, and **body**. Supplementary information: 1. If a third-party gateway is connected, the request address is different from the address displayed by DataArts DataService. In this case, you need to add the **x-forwarded-host** parameter to the request header and set its value to the request address used for signature. 2. If the get request is used, do not define the body.

Error Code	Error Message	Error Cause	Solution
DLG.0902	Fail to call the agent. For details about No matching constant for [-1], see the CDM logs.	The CDM agent cannot be called. 1. The SQL statement duration is too long. 2. CDM resources are insufficient.	1. Check the SQL statement execution duration. If the duration is too long, optimize the SQL statement. (If the default pagination mode is used, you are advised to change it to the custom pagination mode.) 2. If the SQL statement execution duration is short and no other service is running, restart CDM.
DAYU.1088	Failed to process the request sent by the agent.	CDM does not respond.	1. Restart CDM. 2. This issue may be caused by CDM upgrade. Buy another CDM cluster.

1.4 Calling APIs Through App Authentication

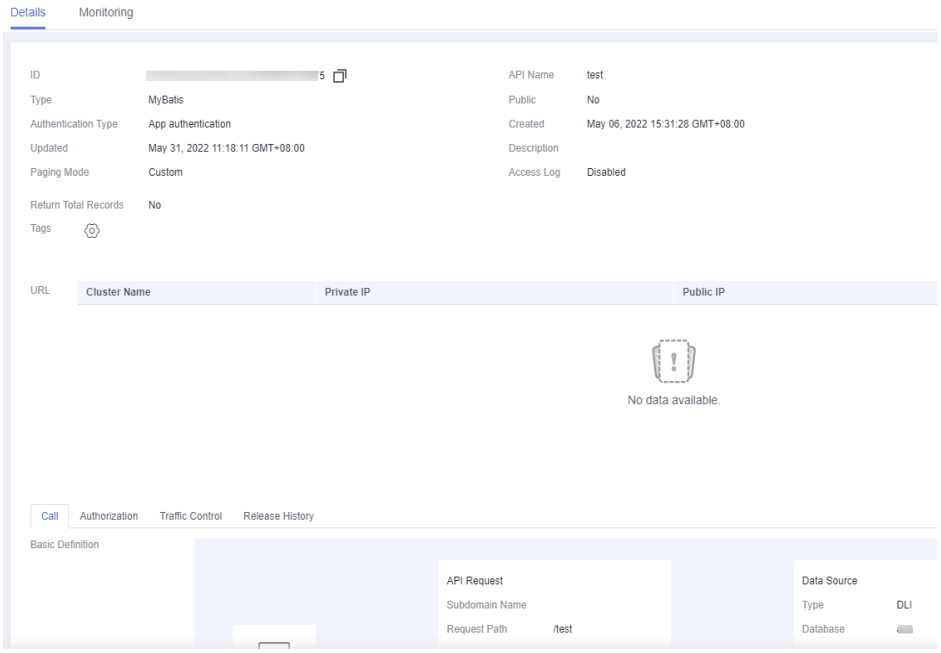
1.4.1 Preparation

Before accessing an API through an SDK in App authentication mode, you must collect the required information.

- Obtain the ID, request URL, and request method of the API.

  On the **API Catalogs** page of DataArts DataService, click an API name. On the **Details** page, view the API ID, request URL, and request method.

Figure 1-1 API details



- Provide a valid AppKey and AppSecret to generate an authentication signature.

  Create an app on the **Apps** page and bind it to the API. Then you can use the AppKey and AppSecert of the app to access the API. View the AppKey and AppSecret on the app details page.

  Figure 1-2 Viewing AppKey and AppSecret

  NOTE

  - AppKey: access key ID of the app. It is the unique ID associated with a secret access key. The AppKey and AppSecret are together used to obtain an encrypted signature for a request.
  - AppSecret: secret access key used together with an AppKey to sign requests. The AppKey and AppSecret can be together used to identify a request sender to prevent the request from being modified.

- When sending an API request, add the current time to the X-Sdk-Date header, and add the signature information to the Authorization header. The signature is valid only within a limited period of time.

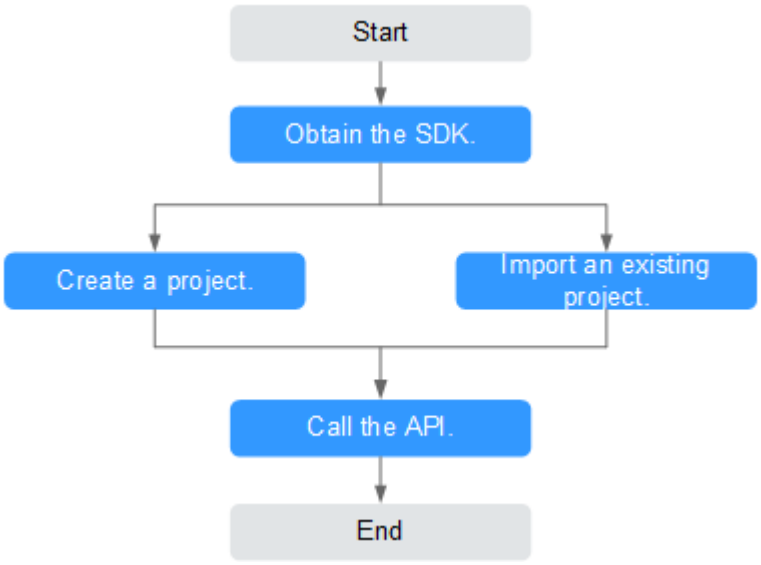
1.4.2 Java

Scenarios

To use Java to call an API through App authentication, obtain the Java SDK, create a project or import an existing project, and then call the API by referring to the API calling example.

This section uses Eclipse 4.5.2 as an example.

Figure 1-3 API calling process



Prerequisites

- You have obtained the domain name, ID, request URL, and request method of the API to be called, and the AppKey and AppSecret of the App for calling the API. For more information, see **Preparation**.
- You have installed Eclipse 3.6.0 or a later version. If not, download Eclipse from the **official Eclipse website** and install it.
- You have installed Java Development Kit (JDK) 1.8.111 or a later version. If not, download JDK from the **official Oracle website** and install it.

Obtaining the SDK

Step 1 Log in to the DataArts Studio console.

Step 2 Click **DataArts DataService**.

Step 3 In the navigation pane, choose **DataArts DataService Exclusive** > **SDKs**.

Step 4 On the **SDKs** page, download the SDK package.

Step 5 Verify integrity of the SDK package. In Windows, open the CLI and run the following command to generate the SHA-256 value of the downloaded SDK package. In the command, **D:\java-sdk.zip** is an example local path and name of the SDK package. Replace it with the actual value.

```
certutil -hashfile D:\java-sdk.zip SHA256
```

The following is an example command output:

```
SHA-256 hash value of D:\java-sdk.zip
a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
CertUtil: -hashfile command executed.
```

Compare the SHA-256 value of the downloaded SDK package with that provided in the following table. If they are the same, no tampering or packet loss occurred during the package download.

Language	SHA-256 Value of the SDK Package
Java	a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
Go	caf22350181a4ecd49dc0d3f56097b10c1794792adae407140950be4ed9b6771
Python	c80b9ef282fc88d3fb16db4cb9d7525d3db71f7989782ed0b636920ea2fadb93
C#	b0e69ef60a561c54c1b86c3447ca855088a1fa2a672fbfba618aaf56b2841e8a
JavaScript	c64e595651de079766e446ce2c3262013256f81683bb9434bee27bed3a4caf01
PHP	e2eba2cae72aea794edb4057ed8eb7cb82f0dbaccabf9c5539694a7a7a9f3c89
C++	c173f59d816aba53f47750cf5ffdc65cc345b1613974b3d2a545ace48787f577
C	e4f22beb7b132fe6e57c9de79f596c3ff830228cd7221b02ca96198e501c642c
Android	d6c3032801ac88cf8cbc51f64d42457174447c8d159f34a187c036913b31ea2b

----End

Obtain the **ApiGateway-java-sdk.zip** package. The following table shows the files decompressed from the package.

Name	Description
libs\	SDK dependencies
libs\java-sdk-core-*x.x.x*.jar	SDK package
src\com\apig\sdk\demo\Main.java	Sample code for signing requests
src\com\apig\sdk\demo\OkHttpDemo.java	
src\com\apig\sdk\demo\LargeFileUploadDemo.java	
.classpath	Java project configuration files
.project	

Importing a Project

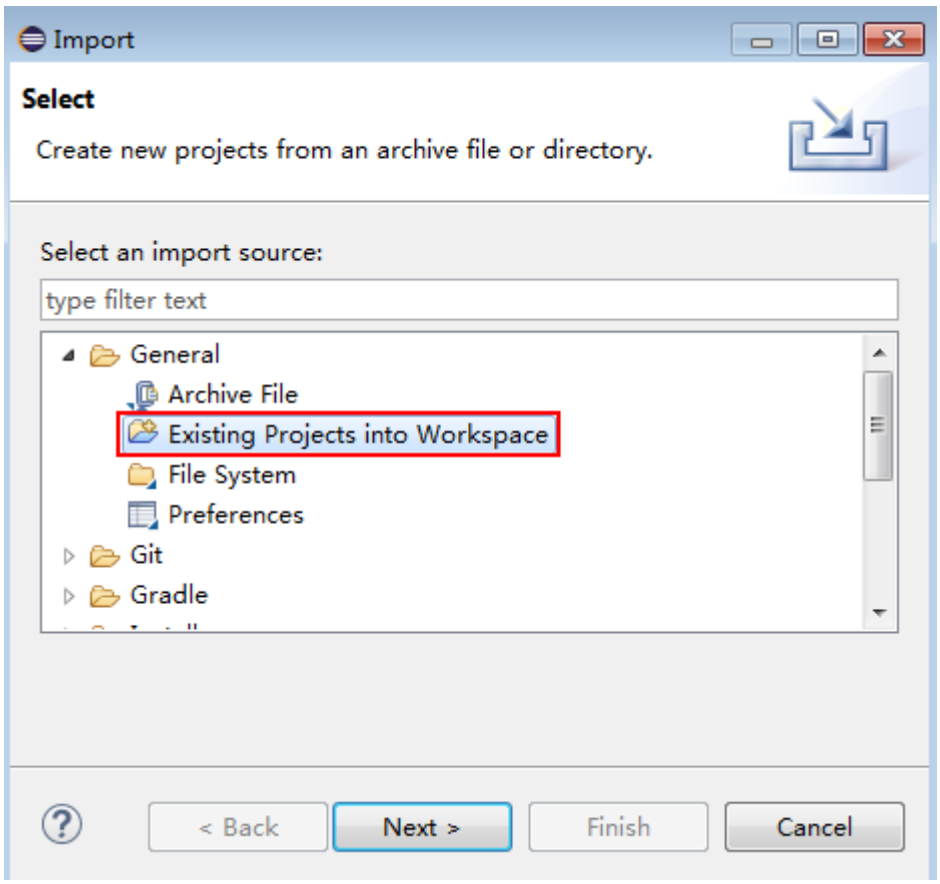
Step 1 Open Eclipse and choose **File** > **Import**.

The **Import** dialog box is displayed.

Step 2 Choose **General** > **Existing Projects into Workspace** and click **Next**.

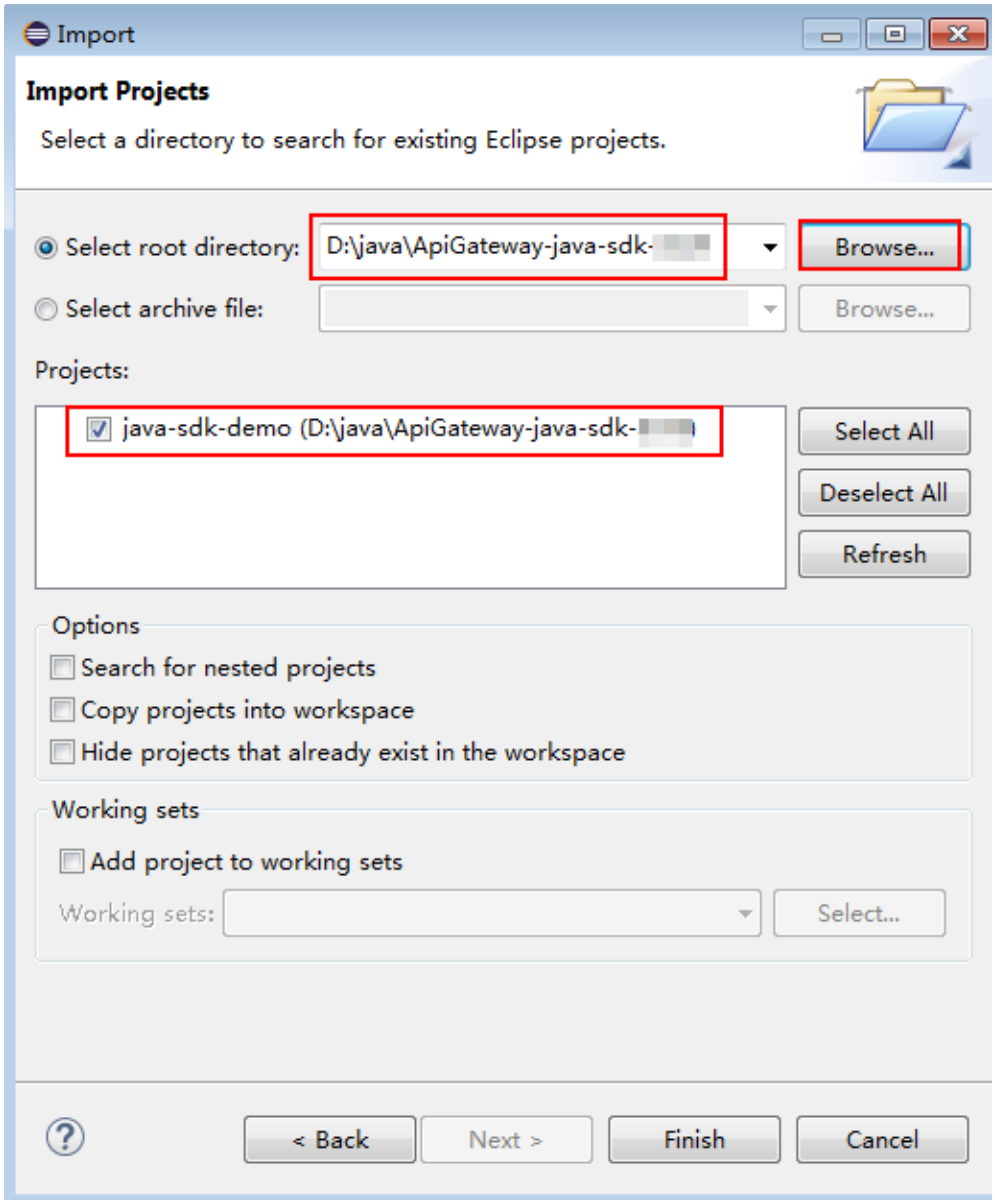
The **Import Projects** dialog box is displayed.

Figure 1-4 Importing a project



Step 3 Click **Browse** and select the directory where the SDK is decompressed.

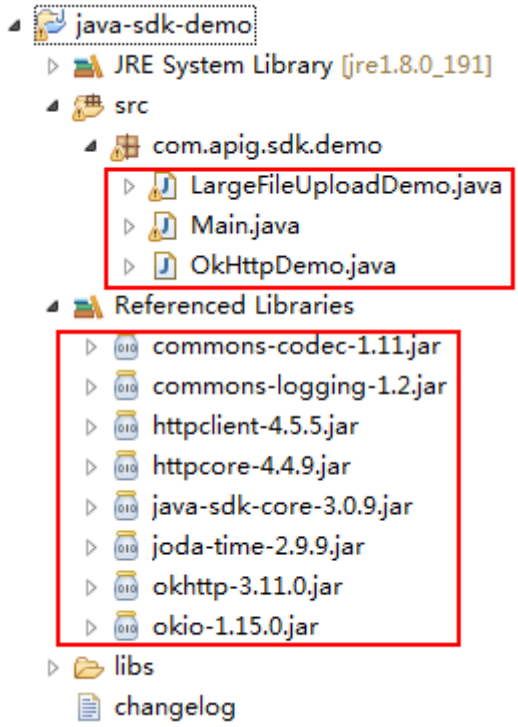
Figure 1-5 Selecting the demo project



Step 4 Click **Finish**.

The following figure shows the directory structure of the project.

Figure 1-6 Directory structure of the imported project



Modify the parameters in sample code **Main.java** as required. For details about the sample code, see **API Calling Example**.

----End

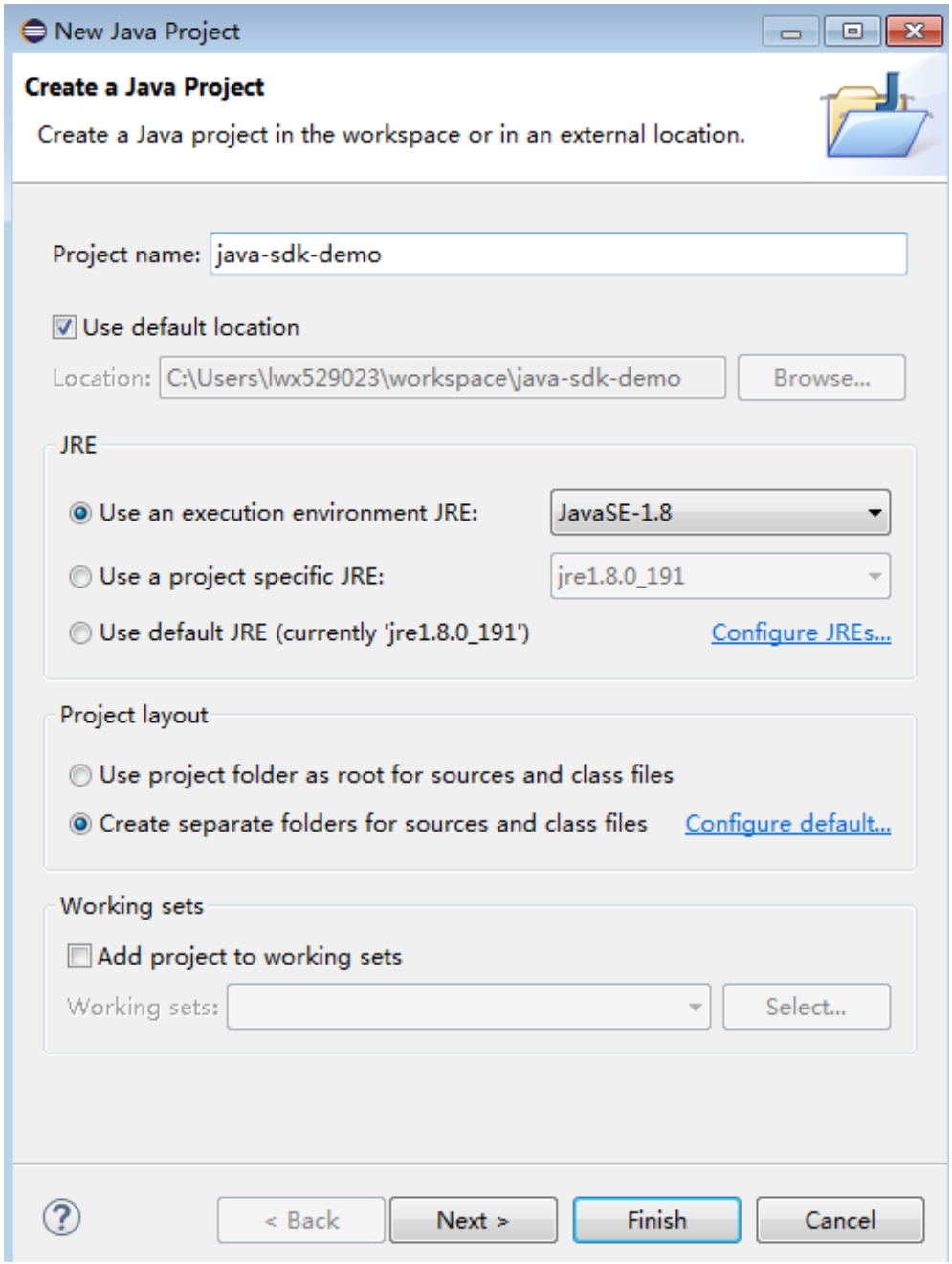
Creating a Project

Step 1 Open Eclipse and choose **File** > **New** > **Java Project**.

The **New Java Project** dialog box is displayed.

Step 2 Enter a project name, for example, **java-sdk-demo**, retain the default settings for other parameters, and click **Finish**.

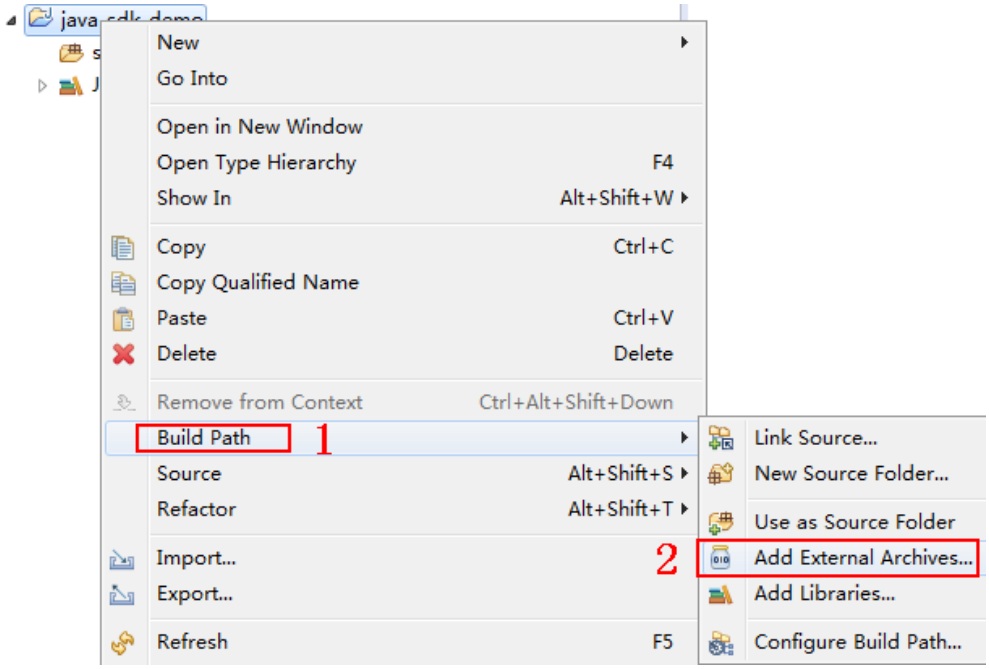
Figure 1-7 Creating a project



Step 3 Import the .jar files in the API Gateway Java SDK.

1. Choose **java-sdk-demo**, right-click, and choose **Build Path** > **Add External Archives** from the shortcut menu.

Figure 1-8 Importing the .jar files



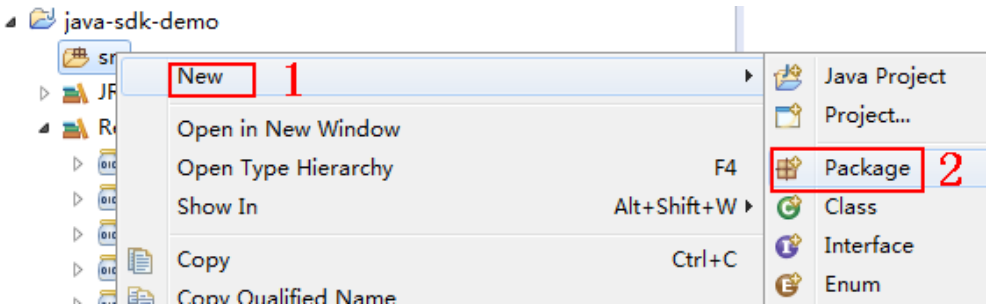
2. Select all .jar files in the **\libs** directory.

Figure 1-9 Selecting all .jar files

Step 4 Create a package and **Main** file.

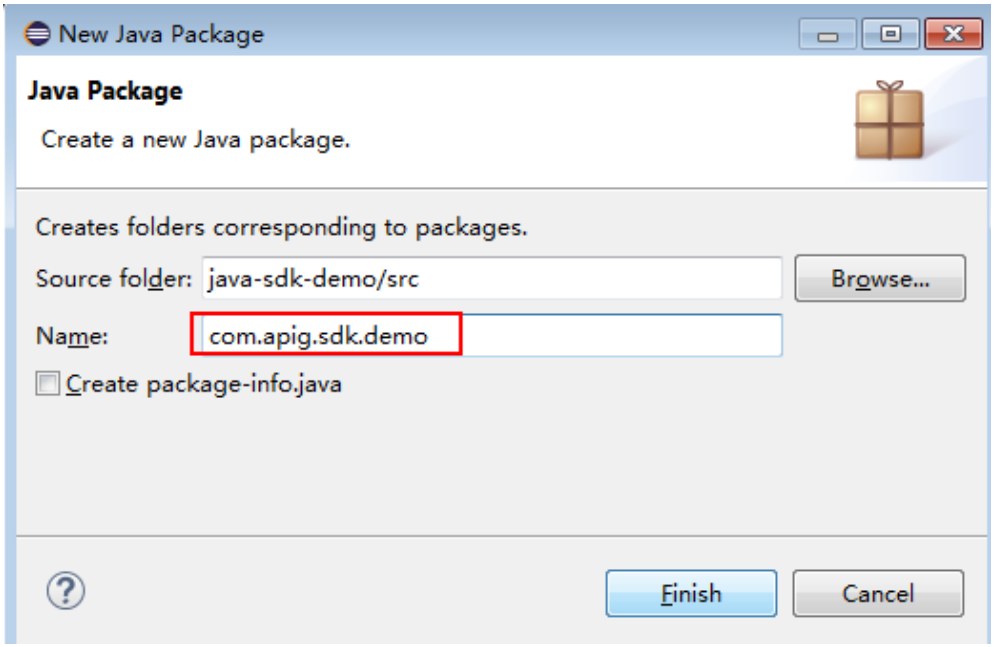
1. Choose **src**, right-click, and choose **New** > **Package** from the shortcut menu.

Figure 1-10 Creating a package



2. Enter **com.apig.sdk.demo** for **Name**.

Figure 1-11 Setting a package name

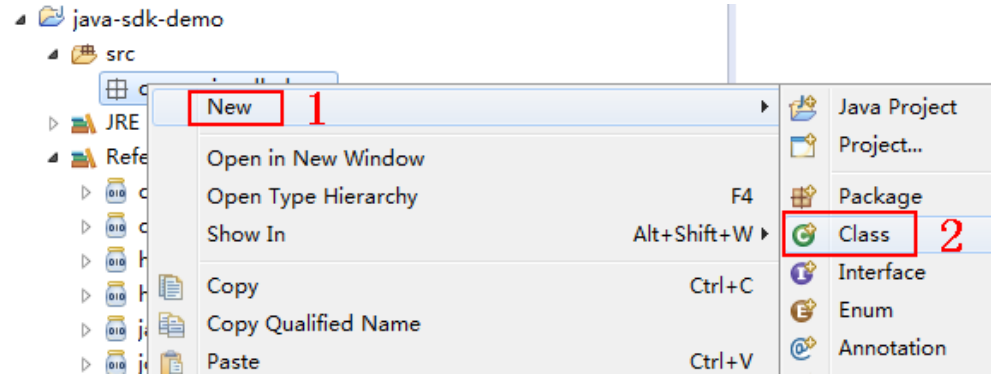


3. Click **Finish**.

   The package is created.

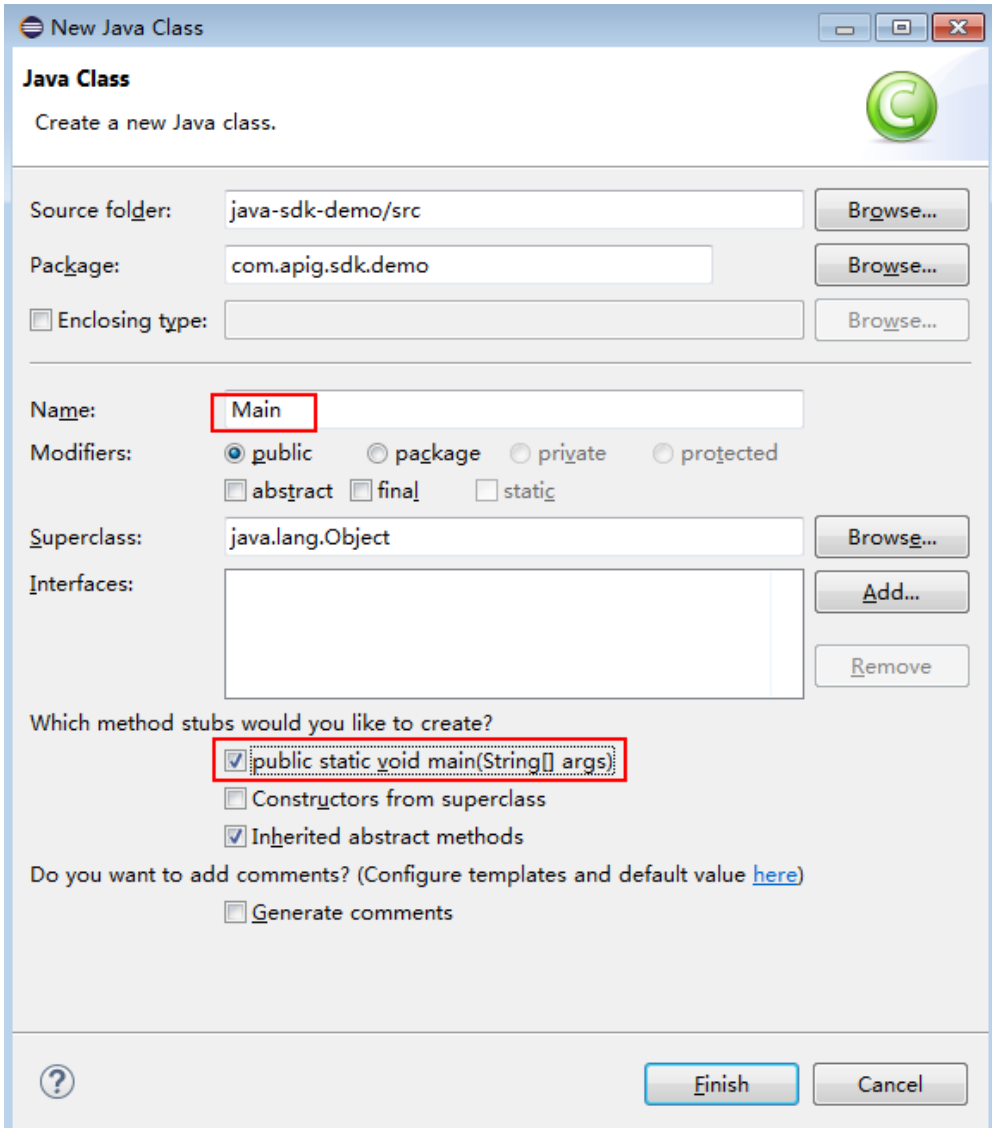
4. Choose **com.apig.sdk.demo**, right-click, and choose **New** > **Class** from the shortcut menu.

Figure 1-12 Creating a class



5. Enter **Main** for **Name** and select **public static void main(String[] args)**.

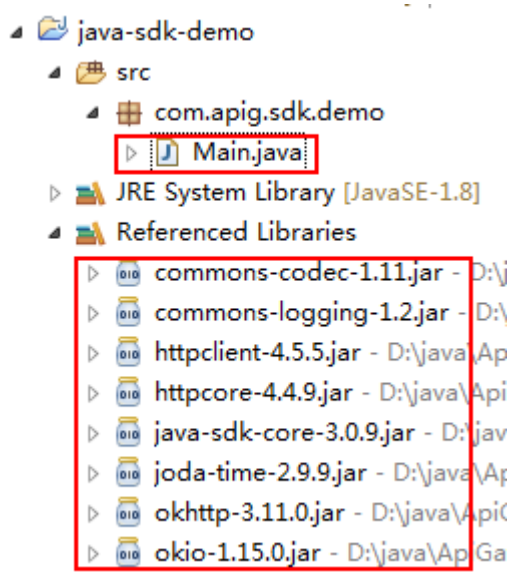
Figure 1-13 Configuring the class



6. Click **Finish**.
   The **Main** file is created.

Step 5 View the directory structure of the project.

Figure 1-14 Directory structure of the new project



Before using **Main.java**, enter the required code according to **API Calling Example**.

----End

API Calling Example

NOTE

- This section demonstrates how to access a published API.
- You need to create and release an API on the DataArts DataService management console. For details about how to create and publish APIs, see *DataArts Studio User Guide*.
- The backend of this API is a fake HTTP service, which returns response code **200** and message body **Congratulations, sdk demo is running**.

Step 1 Add the following references to **Main.java**:

```
import java.io.IOException;
import javax.net.ssl.SSLContext;

import org.apache.http.Header;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
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.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

import com.cloud.apigateway.sdk.utils.Client;
import com.cloud.apigateway.sdk.utils.Request;
```

Step 2 Construct a request by configuring the following parameters:

- **AppKey**: Obtain it by referring to **Preparation**. The sample code uses **4f5f626b-073f-402f-a1e0-e52171c6100c**.

- **AppSecret**: Obtain it by referring to **Preparation**. Set this parameter based on the site requirements. The example code uses ****** as an example.
- **Method**: Specify a request method. The sample code uses **POST**.
- **url**: Request URL of the API, excluding the QueryString and fragment parts. For the domain name, use your own independent domain name bound to the group to which the API belongs. Use the sample code in http://{apig-endpoint}/java-sdk.
- **queryString**: Specify query parameters to be carried in the URL. Characters (0-9a-zA-Z./;[]\-=~#%^&_+:") are allowed. The sample code uses **name=value**.
- Header: Request header. Set a request header as required. It cannot contain underscores (_). The sample code uses **Content-Type:text/plain**.
- **body**: Specify the request body. The sample code uses **demo**.

The sample code is as follows:

```
    Request request = new Request();
    try
    {
        request.setKey("4f5f626b-073f-402f-a1e0-e52171c6100c"); //Obtain the value when creating an app.
        request.setSecret("*****"); //Obtained when an app is created.
        request.setMethod("POST");
        request.setUrl("http://{apig-endpoint}/java-sdk");
         //Obtain the URL when creating an API group.
        request.addQueryStringParam("name", "value");
        request.addHeader("Content-Type", "text/plain");
        //request.addHeader("x-stage", "publish_env_name"); //Specify an environment name before publishing the API in a non-RELEASE environment.
        request.setBody("demo");
    } catch (Exception e)
    {
        e.printStackTrace();
        return;
    }
```

Step 3 Sign the request, add header **x-Authorization**, access the API, and print the result.

The sample code is as follows:

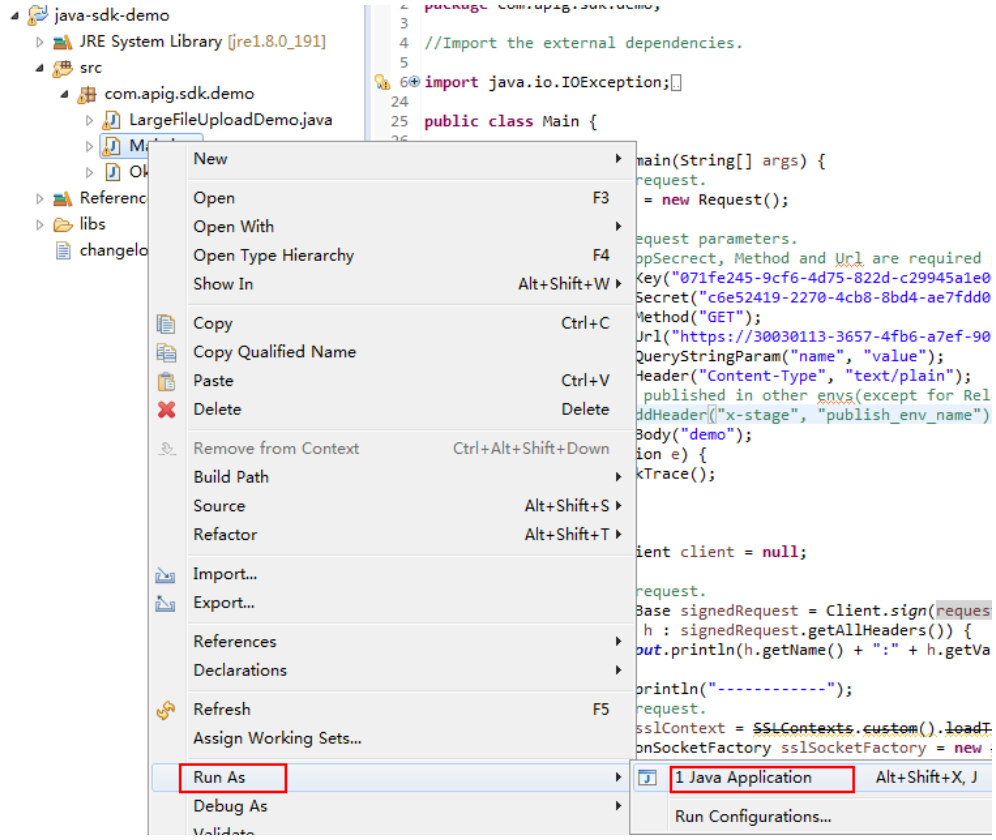
```
    CloseableHttpClient client = null;
    try
    {
        HttpRequestBase signedRequest = Client.sign(request);
        Header[] authorization = signedRequest.getHeaders("Authorization");
        signedRequest.addHeader("x-Authorization",authorization[0].getValue());

        client = HttpClients.custom().build();
        HttpResponse response = client.execute(signedRequest);
        System.out.println(response.getStatusLine().toString());
        Header[] resHeaders = response.getAllHeaders();
        for (Header h : resHeaders)
        {
            System.out.println(h.getName() + ":" + h.getValue());
        }
        HttpEntity resEntity = response.getEntity();
        if (resEntity != null)
        {
            System.out.println(System.getProperty("line.separator") + EntityUtils.toString(resEntity, "UTF-8"));
        }

    } catch (Exception e)
    {
        e.printStackTrace();
    } finally
```



```
    {
        try
        {
            if (client != null)
            {
                client.close();
            }
        } catch (IOException e)
        {
            e.printStackTrace();
        }
    }
```

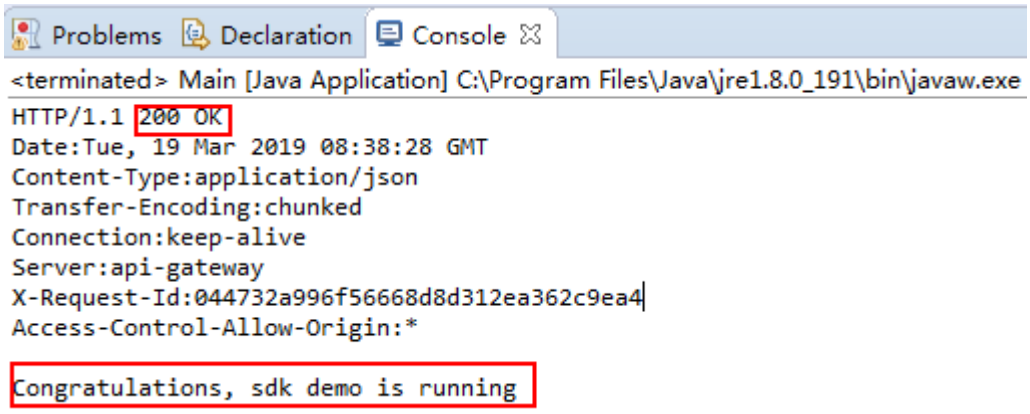
Step 4 Choose **Main.java**, right-click, and choose **Run As** > **Java Application** to run the project test code.

Figure 1-15 Running the project test code



Step 5 On the **Console** tab page, view the running result.

Figure 1-16 Response displayed if the calling is successful



----End

1.4.3 Go

Scenarios

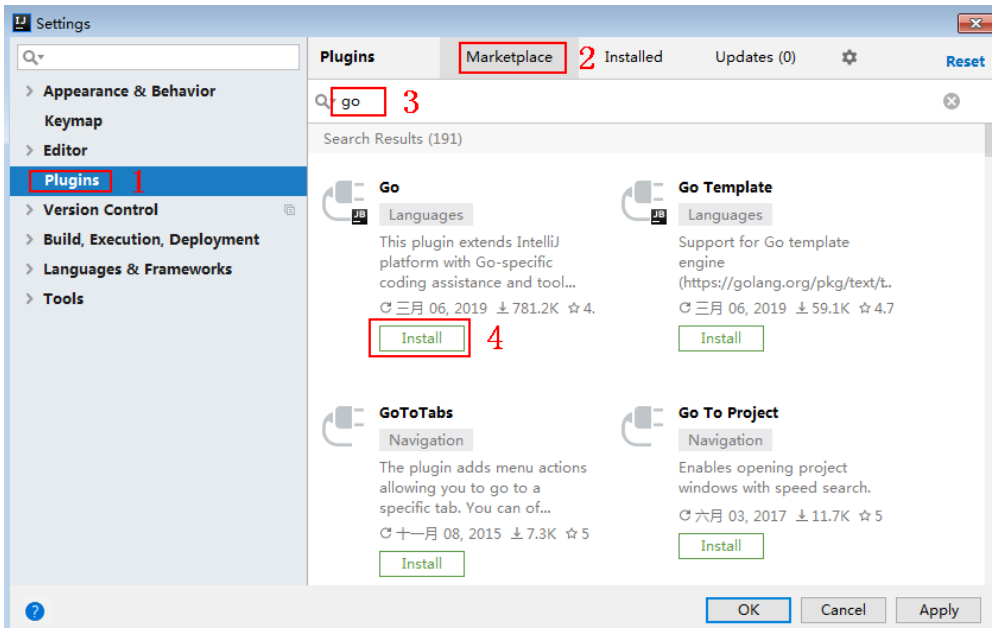
To use Go to call an API through App authentication, obtain the Go SDK, create a project, and then call the API by referring to the API calling example.

This section uses IntelliJ IDEA 2018.3.5 as an example.

Prerequisites

- You have obtained the domain name, ID, request URL, and request method of the API to be called, and the AppKey and AppSecret of the App for calling the API. For more information, see **Preparation**.
- You have installed the Go programming language. If not, download the Go installation package from the **official Go website** and install it.
- You have installed IntelliJ IDEA. If not, download IntelliJ IDEA from the **official IntelliJ IDEA website** and install it.
- You have installed the Go plug-in on IntelliJ IDEA. If not, install the Go plug-in according to **Figure 1-17**.

Figure 1-17 Installing the Go plug-in



Obtaining the SDK

Step 1 Log in to the DataArts Studio console.

Step 2 Click **DataArts DataService**.

Step 3 In the navigation pane, choose **DataArts DataService Exclusive** > **SDKs**.

Step 4 On the **SDKs** page, download the SDK package.

Step 5 Verify integrity of the SDK package. In Windows, open the CLI and run the following command to generate the SHA-256 value of the downloaded SDK package. In the command, **D:\java-sdk.zip** is an example local path and name of the SDK package. Replace it with the actual value.

```
certutil -hashfile D:\java-sdk.zip SHA256
```

The following is an example command output:

```
SHA-256 hash value of D:\java-sdk.zip
a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
CertUtil: -hashfile command executed.
```

Compare the SHA-256 value of the downloaded SDK package with that provided in the following table. If they are the same, no tampering or packet loss occurred during the package download.

Language	SHA-256 Value of the SDK Package
Java	a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
Go	caf22350181a4ecd49dc0d3f56097b10c1794792adae407140950be4ed9b6771

Language	SHA-256 Value of the SDK Package
Python	c80b9ef282fc88d3fb16db4cb9d7525d3db71f7989782ed0b636920ea2fadb93
C#	b0e69ef60a561c54c1b86c3447ca855088a1fa2a672fbfba618aaf56b2841e8a
JavaScript	c64e595651de079766e446ce2c3262013256f81683bb9434bee27bed3a4caf01
PHP	e2eba2cae72aea794edb4057ed8eb7cb82f0dbaccabf9c5539694a7a7a9f3c89
C++	c173f59d816aba53f47750cf5ffdc65cc345b1613974b3d2a545ace48787f577
C	e4f22beb7b132fe6e57c9de79f596c3ff830228cd7221b02ca96198e501c642c
Android	d6c3032801ac88cf8cbc51f64d42457174447c8d159f34a187c036913b31ea2b

----End

Obtain the **ApiGateway-go-sdk.zip** package. The following table shows the files decompressed from the package.

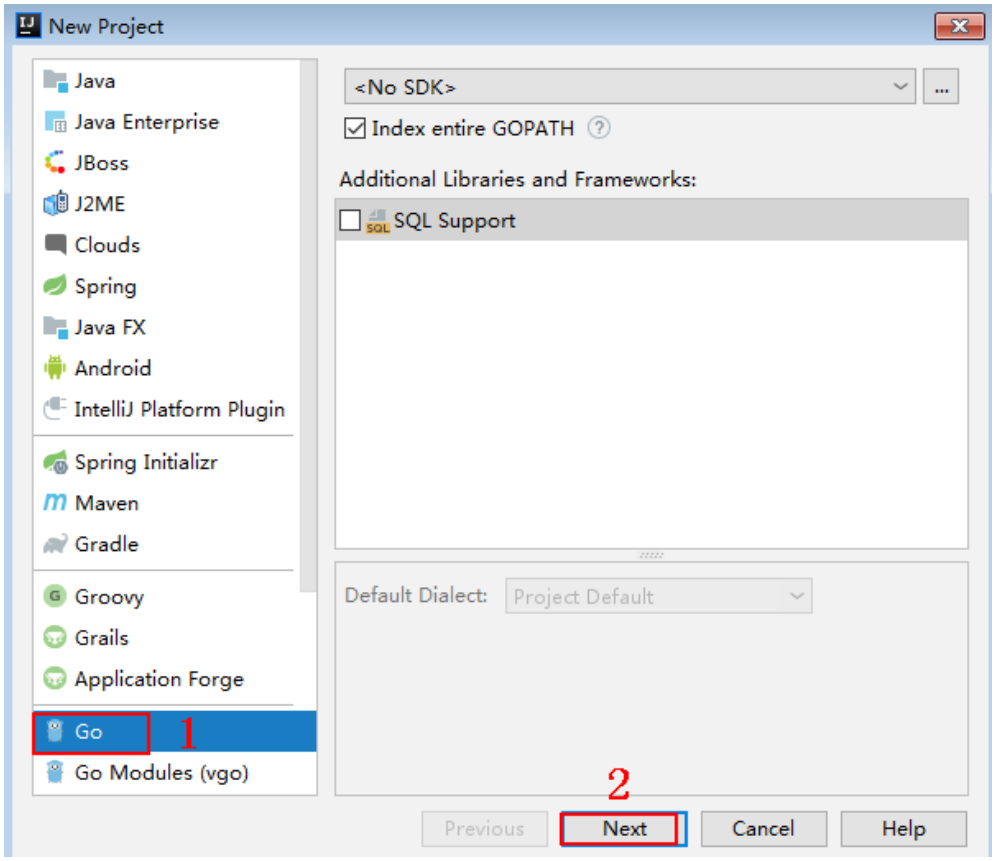
Name	Description
core\escape.go	SDK code
core\signer.go	
demo.go	Sample code

Creating a Project

Step 1 Start IntelliJ IDEA and choose **File** > **New** > **Project**.

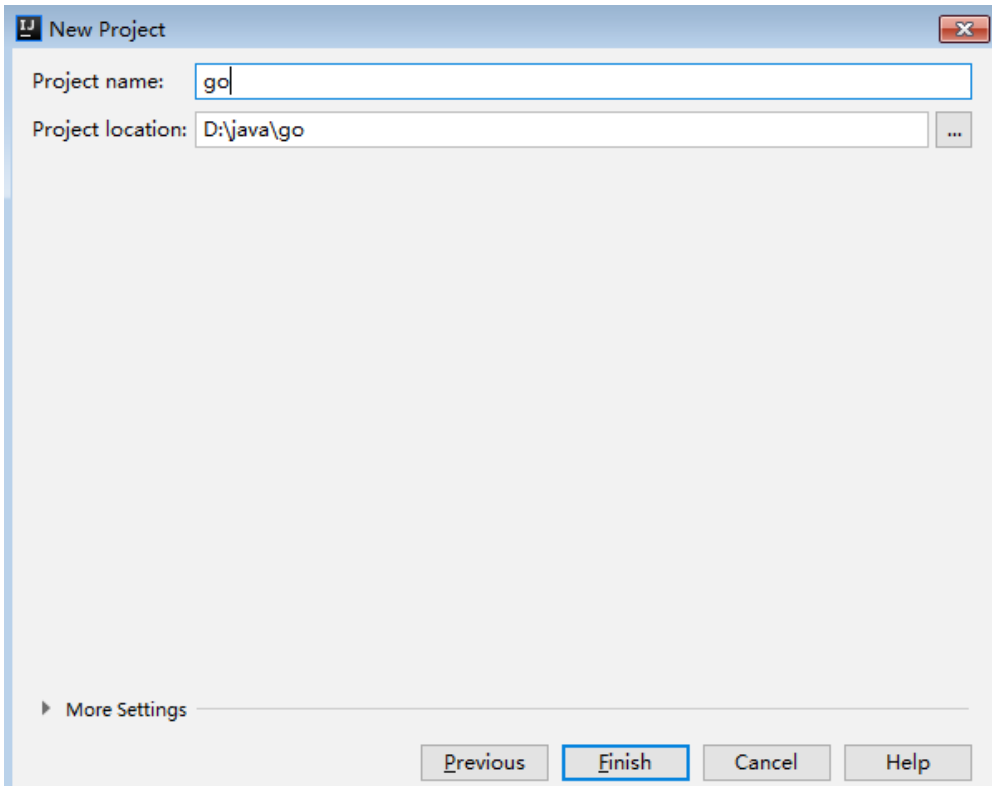
On the displayed **New Project** page, choose **Go** and click **Next**.

Figure 1-18 New Project



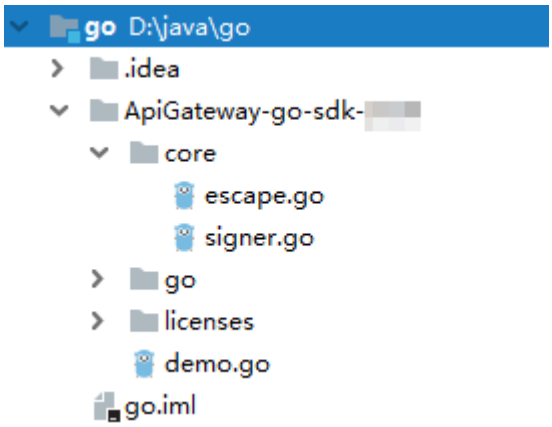
Step 2 Click **...**, select the directory where the SDK is decompressed, and click **Finish**.

Figure 1-19 Selecting the SDK directory after decompression



Step 3 View the directory structure shown in the following figure.

Figure 1-20 Directory structure of the new project



Modify the parameters in sample code **demo.go** as required. For details about the sample code, see **API Calling Example**.

----End

API Calling Example

Step 1 Import the Go SDK (signer.go) to the project.

```
import "apig-sdk/go/core"
```


Step 2 Generate a new signer and enter the AppKey and AppSecret.

```
s := core.Signer{
        Key: "4f5f626b-073f-402f-a1e0-e52171c6100c",
        Secret: "******",
}
```

Step 3 Generate a new request, and specify the domain name, method, request URL, query parameters, and body.

```
r, _ := http.NewRequest("POST", "http:/{apig-endpoint}/api?a=1&b=2",
                        ioutil.NopCloser(bytes.NewBuffer([]byte("foo=bar"))))
```

Step 4 Add a header to the request. The header contains specific parameters. Add other headers to be signed as necessary.

```
r.Header.Add("x-stage", "RELEASE")
r.Header.Add("name","value")
```

Step 5 Execute the following function to add the X-Sdk-Date and Authorization headers for signing: Then, add the **x-Authorization** header to the request. The value of the x-Authorization header is the same as that of the **Authorization** header.

```
s.Sign(r)
authorization := r.Header.Get("Authorization")
r.Header.Add("x-Authorization", authorization)
```

Step 6 Access the API and view the access result.

```
resp, err := http.DefaultClient.Do(r)
body, err := ioutil.ReadAll(resp.Body)
```

----End

1.4.4 Python

Scenarios

To use Python to call an API through App authentication, obtain the Python SDK, create a project, and then call the API by referring to the API calling example.

This section uses IntelliJ IDEA 2018.3.5 as an example.

Preparing the Environment

- You have obtained the domain name, request URL, and request method of the API to be called, and the AppKey and AppSecret of the App for calling the API. For more information, see **Preparation**.
- You have installed Python 2.7.9 or 3.X. If not, download the Python installation package from the **official Python website** and install it.

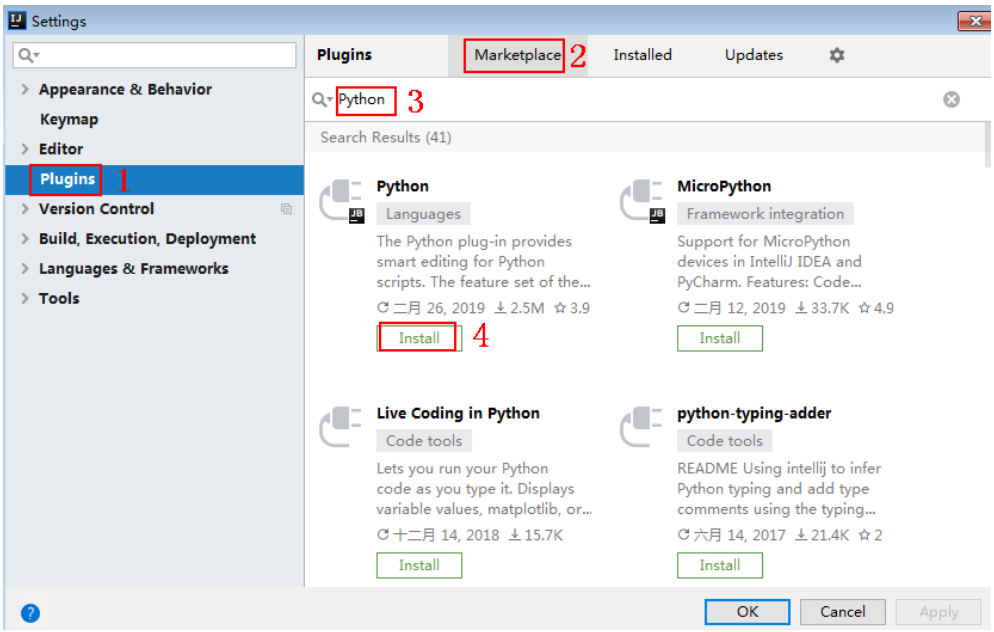
  After Python is installed, run the **pip** command to install the **requests** library.

  ```
  pip install requests
  ```

  NOTE

  If a certificate error occurs during the installation, download the **get-pip.py** file to upgrade the pip environment, and try again.
- You have installed IntelliJ IDEA. If not, download IntelliJ IDEA from the **official IntelliJ IDEA website** and install it.
- You have installed the Python plug-in on IntelliJ IDEA. If not, install the Python plug-in according to **Figure 1-21**.

Figure 1-21 Installing the Python plug-in



Obtaining the SDK

Step 1 Log in to the DataArts Studio console.

Step 2 Click **DataArts DataService**.

Step 3 In the navigation pane, choose **DataArts DataService Exclusive** > **SDKs**.

Step 4 On the **SDKs** page, download the SDK package.

Step 5 Verify integrity of the SDK package. In Windows, open the CLI and run the following command to generate the SHA-256 value of the downloaded SDK package. In the command, **D:\java-sdk.zip** is an example local path and name of the SDK package. Replace it with the actual value.

```
certutil -hashfile D:\java-sdk.zip SHA256
```

The following is an example command output:

```
SHA-256 hash value of D:\java-sdk.zip
a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
CertUtil: -hashfile command executed.
```

Compare the SHA-256 value of the downloaded SDK package with that provided in the following table. If they are the same, no tampering or packet loss occurred during the package download.

Language	SHA-256 Value of the SDK Package
Java	a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
Go	caf22350181a4ecd49dc0d3f56097b10c1794792adae407140950be4ed9b6771

Language	SHA-256 Value of the SDK Package
Python	c80b9ef282fc88d3fb16db4cb9d7525d3db71f7989782ed0b636920ea2fadb93
C#	b0e69ef60a561c54c1b86c3447ca855088a1fa2a672fbfba618aaf56b2841e8a
JavaScript	c64e595651de079766e446ce2c3262013256f81683bb9434bee27bed3a4caf01
PHP	e2eba2cae72aea794edb4057ed8eb7cb82f0dbaccabf9c5539694a7a7a9f3c89
C++	c173f59d816aba53f47750cf5ffdc65cc345b1613974b3d2a545ace48787f577
C	e4f22beb7b132fe6e57c9de79f596c3ff830228cd7221b02ca96198e501c642c
Android	d6c3032801ac88cf8cbc51f64d42457174447c8d159f34a187c036913b31ea2b

----End

Obtain the **ApiGateway-python-sdk.zip** package. The following table shows the files decompressed from the package.

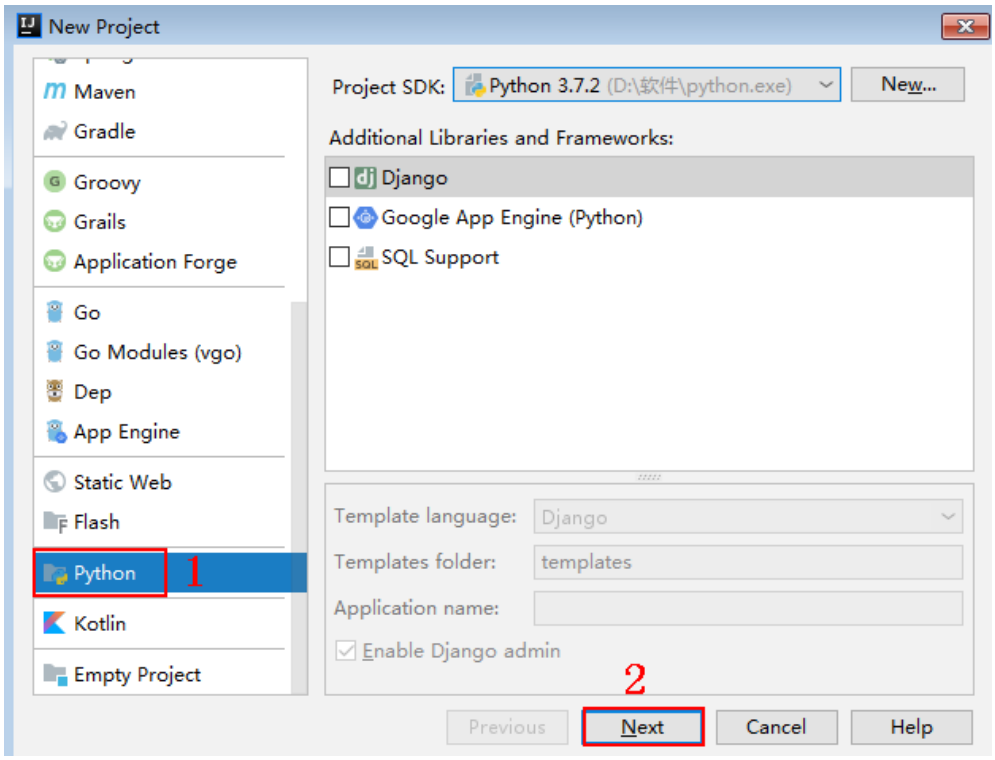
Name	Description
apig_sdk__init__.py	SDK code
apig_sdk\signer.py	
main.py	Sample code
backend_signature.py	Sample code for backend signing
licenses\license-requests	Third-party license

Creating a Project

Step 1 Start IDEA and choose **File** > **New** > **Project**.

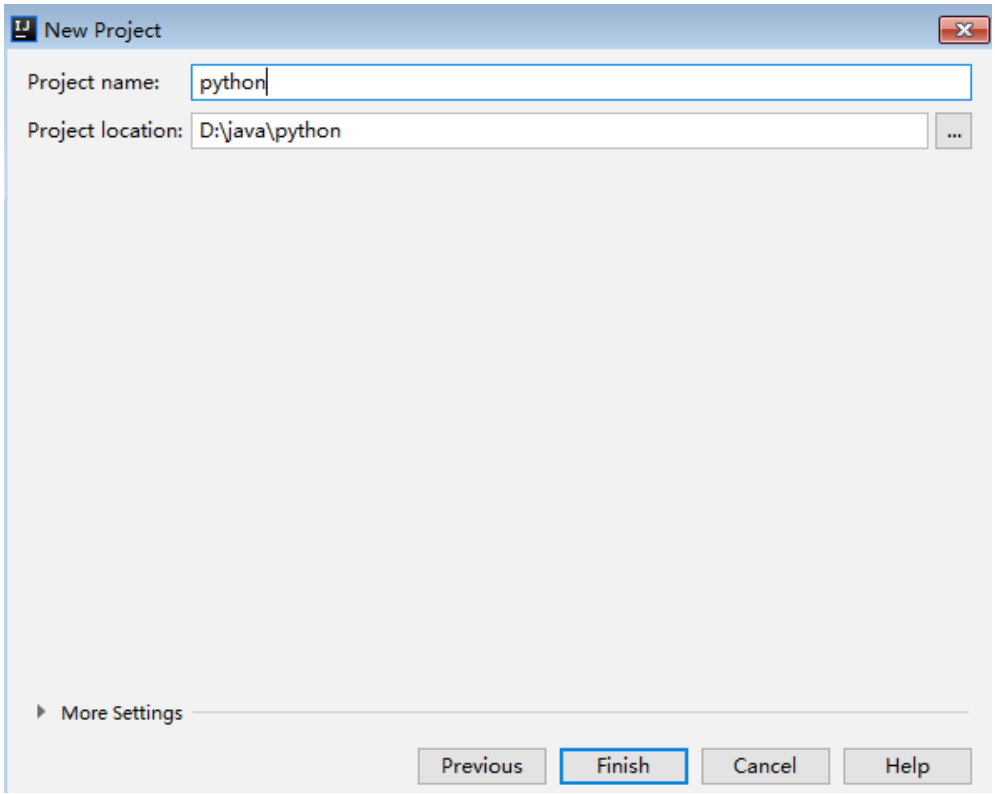
On the displayed **New Project** page, choose **Python** and click **Next**.

Figure 1-22 New Project



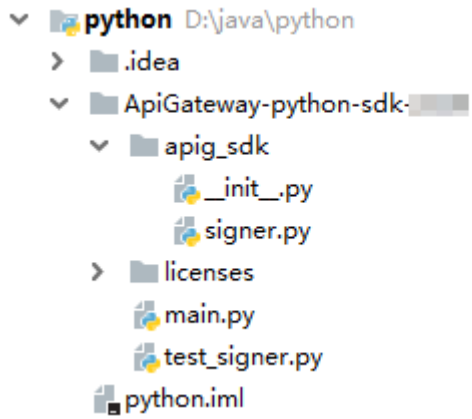
Step 2 Click **Next**. Click **...**, select the directory where the SDK is decompressed, and click **Finish**.

Figure 1-23 Selecting the SDK directory after decompression



Step 3 View the directory structure shown in the following figure.

Figure 1-24 Directory structure of the new project



Modify the parameters in sample code **main.py** as required. For details about the sample code, see **API Calling Example**.

----End

API Calling Example

Step 1 Import **apig_sdk** to the project.

```
from apig_sdk import signer
import requests
```

Step 2 Generate a new signer and enter the AppKey and AppSecret.

```
sig = signer.Signer()
sig.Key = "4f5f626b-073f-402f-a1e0-e52171c6100c"
sig.Secret = "******"
```

Step 3 Generate a request, and specify the method, request URI, header, and request body.

```
r = signer.HttpRequest("POST",
                       "https://{apig-endpoint}/app1?a=1",
                       {"x-stage": "RELEASE", "name": "value"},
                       "body")
```

Step 4 Execute the following function to add the X-Sdk-Date and Authorization headers for signing: Then, add the **x-Authorization** header to the request. The value of the x-Authorization header is the same as that of the **Authorization** header.

```
sig.Sign(r)
r.headers["x-Authorization"] = r.headers["Authorization"]
```

Step 5 Access the API and view the access result.

```
resp = requests.request(r.method, r.scheme + "://" + r.host + r.uri, headers=r.headers, data=r.body)
print(resp.status_code, resp.reason)
print(resp.content)
```

----End

1.4.5 C#

Scenarios

To use C# to call an API through App authentication, obtain the C# SDK, open the project file in the SDK, and then call the API by referring to the API calling example.

Preparing the Environment

- You have obtained the domain name, request URL, and request method of the API to be called, and the AppKey and AppSecret of the App for calling the API. For more information, see **Preparation**.
- You have installed Visual Studio. If not, download it from the **official Visual Studio website** and install it.

Obtaining the SDK

Step 1 Log in to the DataArts Studio console.

Step 2 Click **DataArts DataService**.

Step 3 In the navigation pane, choose **DataArts DataService Exclusive** > **SDKs**.

Step 4 On the **SDKs** page, download the SDK package.

Step 5 Verify integrity of the SDK package. In Windows, open the CLI and run the following command to generate the SHA-256 value of the downloaded SDK package. In the command, **D:\java-sdk.zip** is an example local path and name of the SDK package. Replace it with the actual value.

```
certutil -hashfile D:\java-sdk.zip SHA256
```

The following is an example command output:

```
SHA-256 hash value of D:\java-sdk.zip
a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
CertUtil: -hashfile command executed.
```

Compare the SHA-256 value of the downloaded SDK package with that provided in the following table. If they are the same, no tampering or packet loss occurred during the package download.

Language	SHA-256 Value of the SDK Package
Java	a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
Go	caf22350181a4ecd49dc0d3f56097b10c1794792adae407140950be4ed9b6771
Python	c80b9ef282fc88d3fb16db4cb9d7525d3db71f7989782ed0b636920ea2fadb93
C#	b0e69ef60a561c54c1b86c3447ca855088a1fa2a672fbfba618aaf56b2841e8a

Language	SHA-256 Value of the SDK Package
JavaScript	c64e595651de079766e446ce2c3262013256f81683bb9434bee27bed3a4caf01
PHP	e2eba2cae72aea794edb4057ed8eb7cb82f0dbaccabf9c5539694a7a7a9f3c89
C++	c173f59d816aba53f47750cf5ffdc65cc345b1613974b3d2a545ace48787f577
C	e4f22beb7b132fe6e57c9de79f596c3ff830228cd7221b02ca96198e501c642c
Android	d6c3032801ac88cf8cbc51f64d42457174447c8d159f34a187c036913b31ea2b

----End

Obtain the **ApiGateway-csharp-sdk.zip** package. The following table shows the files decompressed from the package.

Name	Description
apigateway-signature\Signer.cs	SDK code
apigateway-signature\HttpEncoder.cs	
sdk-request\Program.cs	Sample code for signing requests
backend-signature\	Sample project for backend signing
csharp.sln	Project file
licenses\license-referencesource	Third-party license

Opening a Project

Double-click **csharp.sln** in the SDK package to open the project. The project contains the following:

- **apigateway-signature**: Shared library that implements the signature algorithm. It can be used in the .Net Framework and .Net Core projects.
- **backend-signature**: Example of a backend service signature.
- **sdk-request**: Example of invoking the signature algorithm. Modify the parameters as required. For details about the sample code, see **API Calling Example**.

API Calling Example

Step 1 Import the C# SDK to your project.

```
using APIGATEWAY_SDK;
```

Step 2 Generate a new signer and enter the AppKey and AppSecret.

```
Signer signer = new Signer();
signer.Key = "4f5f626b-073f-402f-a1e0-e52171c6100c";
signer.Secret = "******";
```

Step 3 Generate an HttpRequest, and specify the method, request URL, and body.

```
HttpRequest r = new HttpRequest("POST",
                new Uri("https://{apig-endpoint}/app1?query=value"));
r.body = "{\"a\":1}";
```

Step 4 Add a header to the request. The header contains specific parameters. Add other headers to be signed as necessary.

```
r.headers.Add("x-stage", "RELEASE");
r.headers.Add("name","value");
```

Step 5 Execute the following function to generate **HttpWebRequest**, and add the X-Sdk-Date and Authorization headers for signing the request: Then, add the **x-Authorization** header to the request. The value of the x-Authorization header is the same as that of the **Authorization** header.

```
HttpWebRequest req = signer.Sign(r);
req.Headers.Add("x-Authorization", string.Join(", ", req.Headers.GetValues("x-Authorization")));
```

Step 6 Access the API and view the access result.

```
var writer = new StreamWriter(req.GetRequestStream());
writer.Write(r.body);
writer.Flush();
HttpWebResponse resp = (HttpWebResponse)req.GetResponse();
var reader = newStreamReader(resp.GetResponseStream());
Console.WriteLine(reader.ReadToEnd());
```

----End

1.4.6 JavaScript

Scenarios

To use JavaScript to call an API through App authentication, obtain the JavaScript SDK, create a new project, and then call the API by referring to the API calling example.

This section uses IntelliJ IDEA 2018.3.5 as an example to describe how to set up a Node.js development environment.

Preparing the Environment

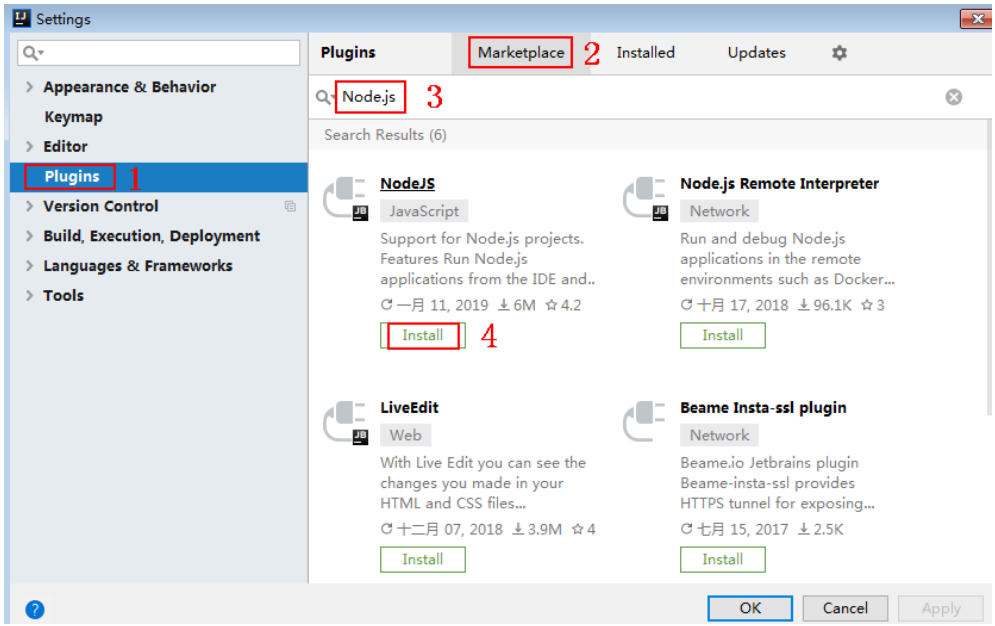
- You have obtained the domain name, request URL, and request method of the API to be called, and the AppKey and AppSecret of the App for calling the API. For more information, see **Preparation**.
- You have installed the Node.js programming language. If not, download the Node.js installation package from the **official Node.js website** and install it.

  After Node.js is installed, run the **npm** command to install the **moment** and **moment-timezone** modules.


```
npm install moment --save
npm install moment-timezone --save
```

- You have installed IntelliJ IDEA. If not, download IntelliJ IDEA from the **official IntelliJ IDEA website** and install it.
- You have installed the Node.js plug-in on IntelliJ IDEA. If not, install the Python plug-in according to **Figure 1-25**.

Figure 1-25 Installing the Node.js plug-in



Obtaining the SDK

Step 1 Log in to the DataArts Studio console.

Step 2 Click **DataArts DataService**.

Step 3 In the navigation pane, choose **DataArts DataService Exclusive** > **SDKs**.

Step 4 On the **SDKs** page, download the SDK package.

Step 5 Verify integrity of the SDK package. In Windows, open the CLI and run the following command to generate the SHA-256 value of the downloaded SDK package. In the command, **D:\java-sdk.zip** is an example local path and name of the SDK package. Replace it with the actual value.

```
certutil -hashfile D:\java-sdk.zip SHA256
```

The following is an example command output:

```
SHA-256 hash value of D:\java-sdk.zip
a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
CertUtil: -hashfile command executed.
```

Compare the SHA-256 value of the downloaded SDK package with that provided in the following table. If they are the same, no tampering or packet loss occurred during the package download.

Language	SHA-256 Value of the SDK Package
Java	a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
Go	caf22350181a4ecd49dc0d3f56097b10c1794792adae407140950be4ed9b6771
Python	c80b9ef282fc88d3fb16db4cb9d7525d3db71f7989782ed0b636920ea2fadb93
C#	b0e69ef60a561c54c1b86c3447ca855088a1fa2a672fbfba618aaf56b2841e8a
JavaScript	c64e595651de079766e446ce2c3262013256f81683bb9434bee27bed3a4caf01
PHP	e2eba2cae72aea794edb4057ed8eb7cb82f0dbaccabf9c5539694a7a7a9f3c89
C++	c173f59d816aba53f47750cf5ffdc65cc345b1613974b3d2a545ace48787f577
C	e4f22beb7b132fe6e57c9de79f596c3ff830228cd7221b02ca96198e501c642c
Android	d6c3032801ac88cf8cbc51f64d42457174447c8d159f34a187c036913b31ea2b

----End

Obtain the **ApiGateway-javascript-sdk.zip** package. The following table shows the files decompressed from the package.

Name	Description
signer.js	SDK code
node_demo.js	Node.js sample code
demo.html	Browser sample code
demo_require.html	Browser sample code (loaded using **require**)
test.js	Test case
js\hmac-sha256.js	Dependencies
js\moment.min.js	
js\moment-timezone-with-data.min.js	
licenses\license-crypto-js	Third-party licenses
licenses\license-moment	

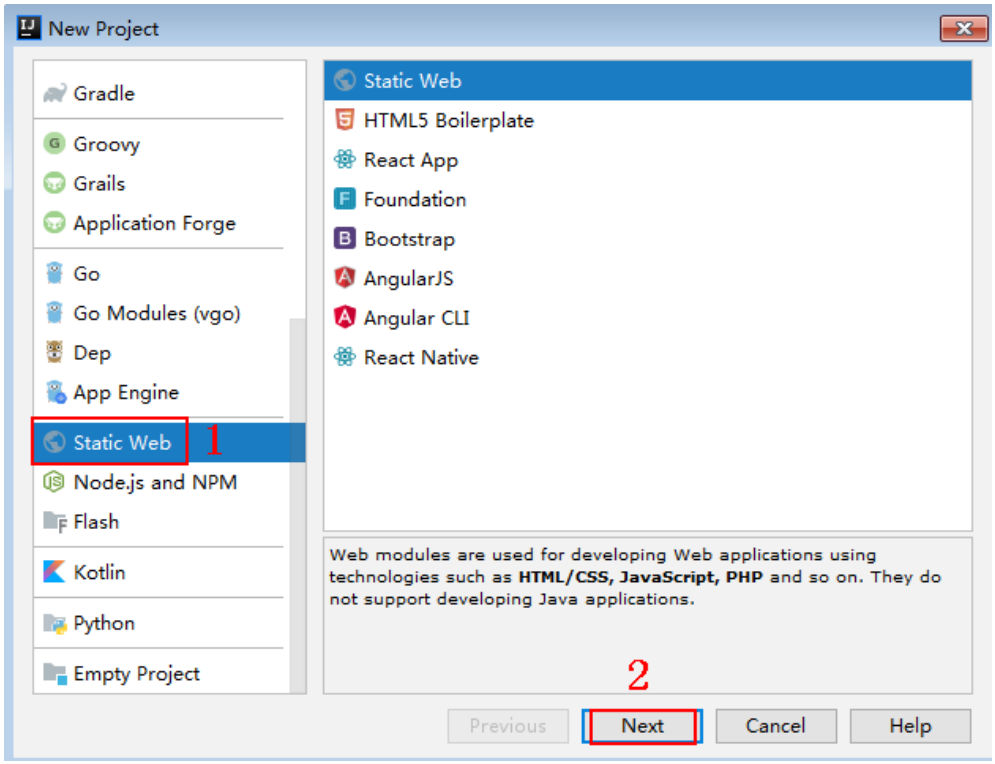
Name	Description
licenses\license-moment-timezone	
licenses\license-node	

Creating a Project

Step 1 Start IntelliJ IDEA and choose **File** > **New** > **Project**.

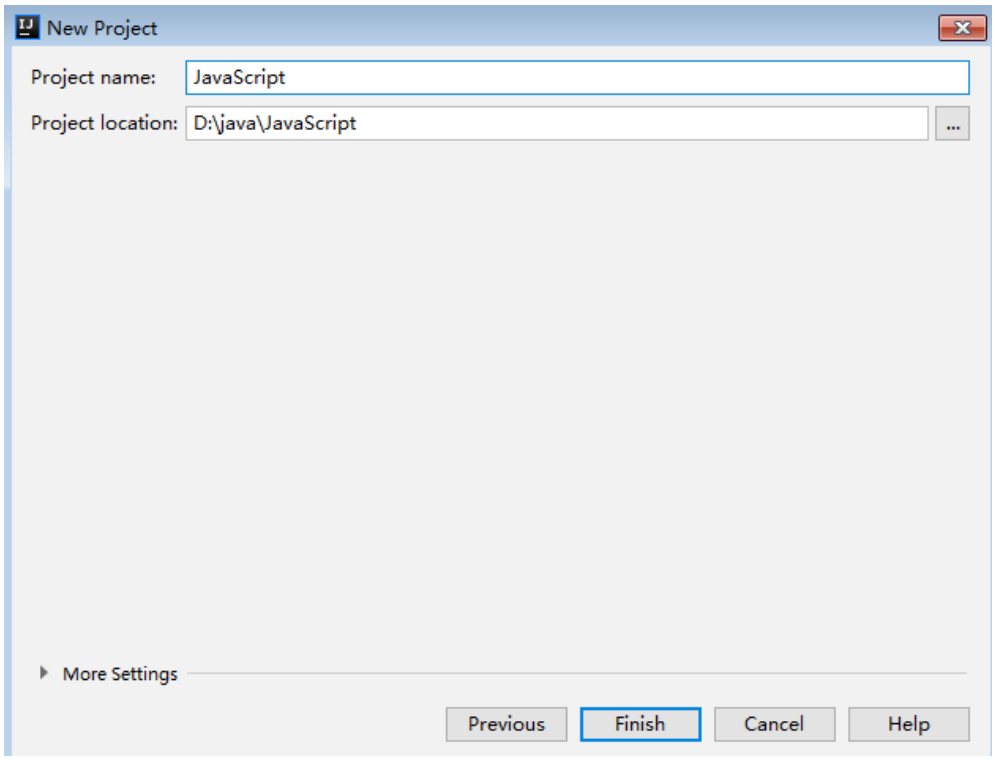
In the **New Project** dialog box, choose **Static Web** and click **Next**.

Figure 1-26 New Project



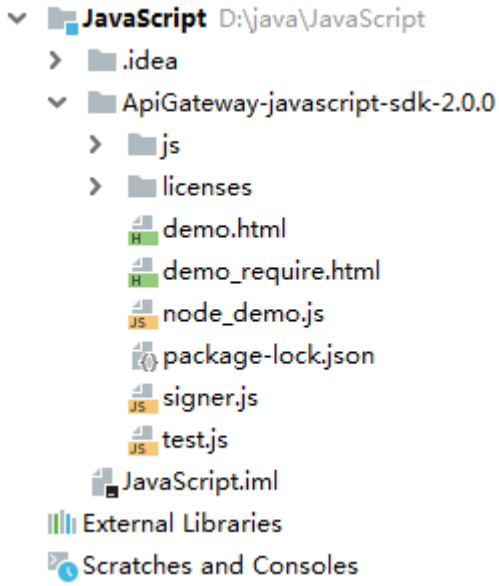
Step 2 Click **...**, select the directory where the SDK is decompressed, and click **Finish**.

Figure 1-27 Selecting the SDK directory after decompression



Step 3 View the directory structure shown in the following figure.

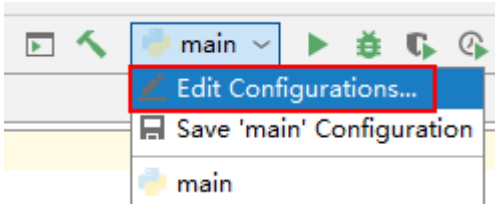
Figure 1-28 Directory structure of the new project



- **node_demo.js**: Sample code in Node.js. Modify the parameters in the sample code as required. For details about the sample code, see **API Calling Example (Node.js)**.

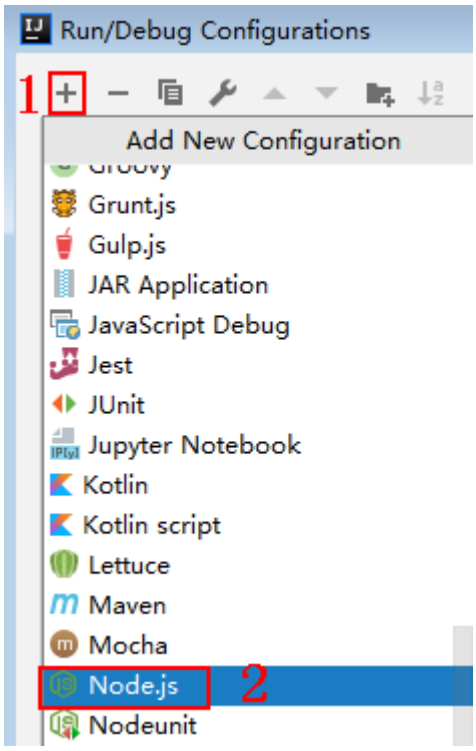
Step 4 Click **Edit Configurations**.

Figure 1-29 Edit Configurations



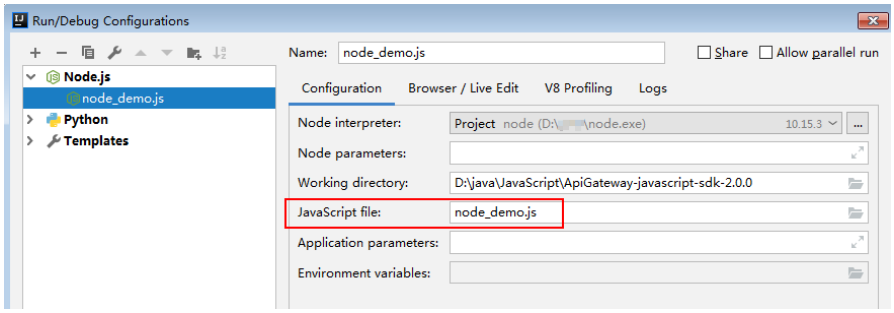
Step 5 Click + and select **Node.js**.

Figure 1-30 Selecting Node.js



Step 6 Set **JavaScript file** to **node_demo.js** and click **OK**.

Figure 1-31 Selecting node_demo.js



----End

API Calling Example (Node.js)

Step 1 Import **signer.js** to your project.

```
var signer = require('./signer')
var http = require('http')
```

Step 2 Generate a new signer and enter the AppKey and AppSecret.

```
var sig = new signer.Signer()
sig.Key = "4f5f626b-073f-402f-a1e0-e52171c6100c"
sig.Secret = "******"
```

Step 3 Generate a request, and specify the method, request URI, and request body.

```
var r = new signer.HttpRequest("POST", "{apig-endpoint}/app1?a=1");
r.body = '{"a":1}'
```

Step 4 Add a header to the request. The header contains specific parameters. Add other headers to be signed as necessary.

```
r.headers = { "x-stage":"RELEASE", "name":"value"}
```

Step 5 Execute the following function to generate HTTP(s) request parameters, and add the X-Sdk-Date and Authorization headers for signing the request: Then, add the **x-Authorization** header to the request. The value of the x-Authorization header is the same as that of the **Authorization** header.

```
var opt = sig.Sign(r)
opt.headers["x-Authorization"] = opt.headers["Authorization"]
```

Step 6 Access the API and view the access result. If you access the API using HTTPS, change **http.request** to **https.request**.

```
var req=http.request(opt, function(res){
        console.log(res.statusCode)
        res.on("data",    function(chunk){
        console.log(chunk.toString())
    })
})
req.on("error",function(err){
    console.log(err.message)
})
req.write(r.body)
req.end()
```

----End

1.4.7 PHP

Scenarios

To use PHP to call an API through App authentication, obtain the PHP SDK, create a new project, and then call the API by referring to **API Calling Example**.

This section uses IntelliJ IDEA 2018.3.5 as an example.

Preparing the Environment

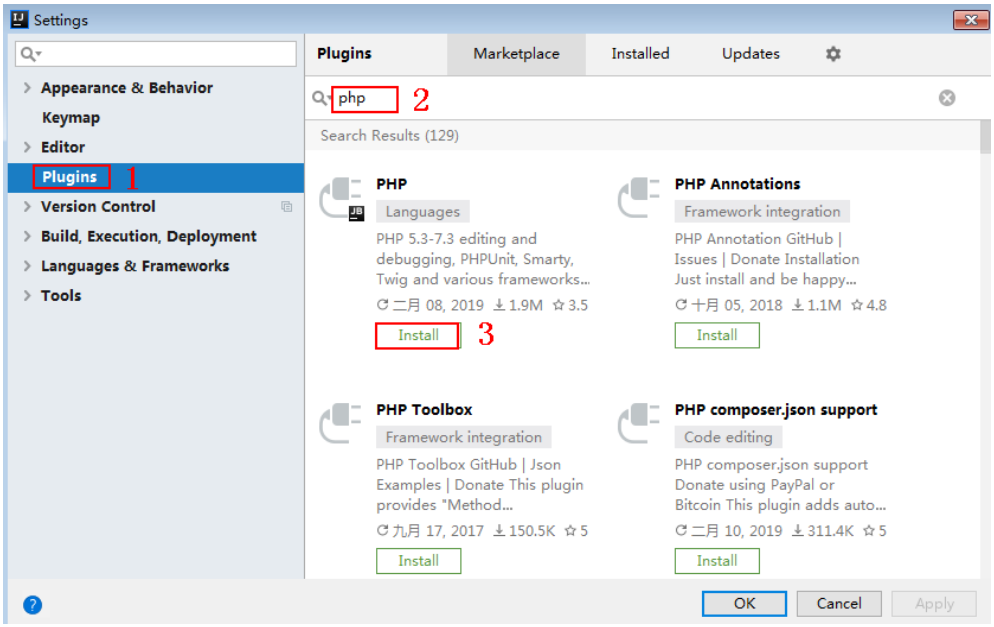
- You have obtained the domain name, request URL, and request method of the API to be called, and the AppKey and AppSecret of the App for calling the API. For more information, see **Preparation**.
- You have installed IntelliJ IDEA. If not, download IntelliJ IDEA from the **official IntelliJ IDEA website** and install it.

- You have installed the PHP programming language. If not, download the PHP installation package from the **official PHP website** and install it.
- Copy the **php.ini-production** file from the PHP installation directory to the **C:\windows** directory, rename the file as **php.ini**, and then add the following lines to the file:

```
extension_dir = "PHP installation directory/ext"
extension=openssl
extension=curl
```

- You have installed the PHP plug-in on IntelliJ IDEA. If not, install the PHP plug-in according to **Figure 1-32**.

Figure 1-32 Installing the PHP plug-in



Obtaining the SDK

Step 1 Log in to the DataArts Studio console.

Step 2 Click **DataArts DataService**.

Step 3 In the navigation pane, choose **DataArts DataService Exclusive** > **SDKs**.

Step 4 On the **SDKs** page, download the SDK package.

Step 5 Verify integrity of the SDK package. In Windows, open the CLI and run the following command to generate the SHA-256 value of the downloaded SDK package. In the command, **D:\java-sdk.zip** is an example local path and name of the SDK package. Replace it with the actual value.

```
certutil -hashfile D:\java-sdk.zip SHA256
```

The following is an example command output:

```
SHA-256 hash value of D:\java-sdk.zip
a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
CertUtil: -hashfile command executed.
```

Compare the SHA-256 value of the downloaded SDK package with that provided in the following table. If they are the same, no tampering or packet loss occurred during the package download.

Language	SHA-256 Value of the SDK Package
Java	a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
Go	caf22350181a4ecd49dc0d3f56097b10c1794792adae407140950be4ed9b6771
Python	c80b9ef282fc88d3fb16db4cb9d7525d3db71f7989782ed0b636920ea2fadb93
C#	b0e69ef60a561c54c1b86c3447ca855088a1fa2a672fbfba618aaf56b2841e8a
JavaScript	c64e595651de079766e446ce2c3262013256f81683bb9434bee27bed3a4caf01
PHP	e2eba2cae72aea794edb4057ed8eb7cb82f0dbaccabf9c5539694a7a7a9f3c89
C++	c173f59d816aba53f47750cf5ffdc65cc345b1613974b3d2a545ace48787f577
C	e4f22beb7b132fe6e57c9de79f596c3ff830228cd7221b02ca96198e501c642c
Android	d6c3032801ac88cf8cbc51f64d42457174447c8d159f34a187c036913b31ea2b

----End

Obtain the **ApiGateway-php-sdk.zip** package. The following table shows the files decompressed from the package.

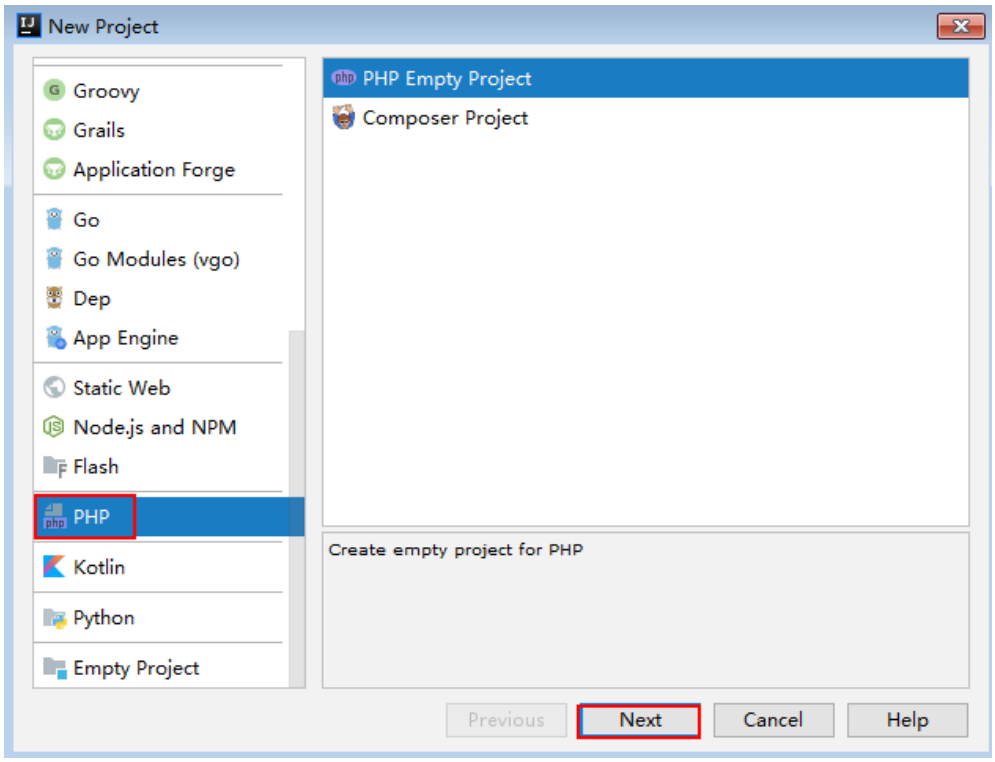
Name	Description
signer.php	SDK code
index.php	Sample code

Creating a Project

Step 1 Start IDEA and choose **File** > **New** > **Project**.

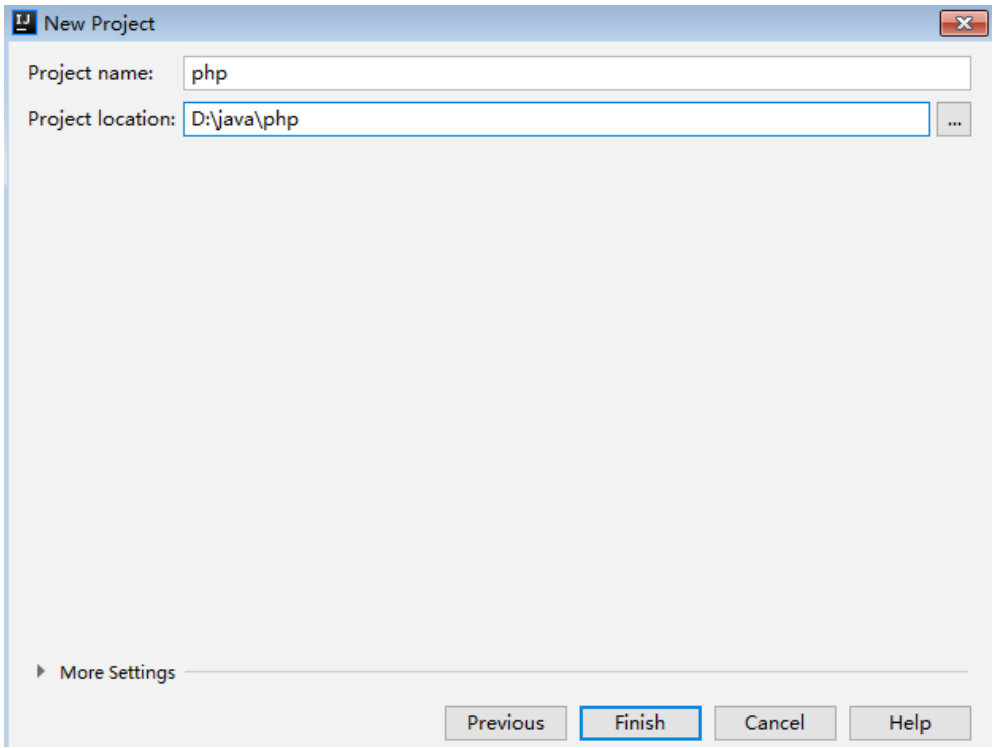
On the displayed **New Project** page, choose **PHP** and click **Next**.

Figure 1-33 New Project



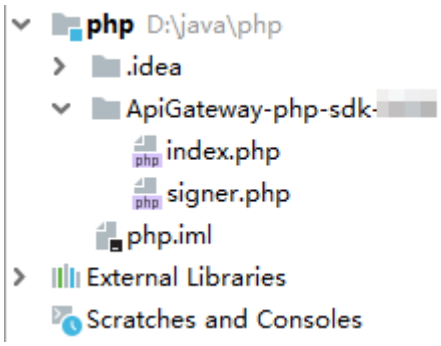
Step 2 Click **...**, select the directory where the SDK is decompressed, and click **Finish**.

Figure 1-34 Selecting the SDK directory after decompression



Step 3 View the directory structure shown in the following figure.

Figure 1-35 Directory structure of the new project



Modify the parameters in sample code **signer.php** as required. For details about the sample code, see **API Calling Example**.

----End

API Calling Example

Step 1 Import the PHP SDK to your code.

```
require 'signer.php';
```

Step 2 Generate a new signer and enter the AppKey and AppSecret.

```
$signer = new Signer();
$signer->Key = '4f5f626b-073f-402f-a1e0-e52171c6100c';
$signer->Secret = "******";
```

Step 3 Generate a new request, and specify the method, request URL, and body.

```
$req = new Request('GET', "https://{apig-endpoint}/app1?a=1");
$req->body = '';
```

Step 4 Add a header to the request. The header contains specific parameters. Add other headers to be signed as necessary. **host** is mandatory. Enter the request URL. The following is an example:

```
$req->headers = array(
    'host' => '{apig-endpoint}'
);
```

Step 5 Execute the following function to generate a **$curl** context variable. Then, add the **x-Authorization** header to the request. The value of the x-Authorization header is the same as that of the **Authorization** header.

```
$curl = $signer->Sign($req);
$req->headers['x-Authorization'] = $req->headers['Authorization'];
$header = array();
foreach ($req->headers as $key => $value) {
    array_push($header, strtolower($key) . ':' . trim($value));
}
curl_setopt($curl, CURLOPT_HTTPHEADER, $header);
```

Step 6 Access the API and view the access result.

```
$response = curl_exec($curl);
echo curl_getinfo($curl, CURLINFO_HTTP_CODE);
echo $response;
curl_close($curl);
```

----End

1.4.8 C++

Scenarios

To use C++ to call an API through App authentication, obtain the C++ SDK, and then call the API by referring to the API calling example.

Preparing the Environment

1. You have obtained the domain name, request URL, and request method of the API to be called, and the AppKey and AppSecret of the App for calling the API. For more information, see **Preparation**.
2. Install the OpenSSL library.
   ```
   apt-get install libssl-dev
   ```
3. Install the curl library.
   ```
   apt-get install libcurl4-openssl-dev
   ```

Obtaining the SDK

Step 1 Log in to the DataArts Studio console.

Step 2 Click **DataArts DataService**.

Step 3 In the navigation pane, choose **DataArts DataService Exclusive** > **SDKs**.

Step 4 On the **SDKs** page, download the SDK package.

Step 5 Verify integrity of the SDK package. In Windows, open the CLI and run the following command to generate the SHA-256 value of the downloaded SDK package. In the command, **D:\java-sdk.zip** is an example local path and name of the SDK package. Replace it with the actual value.

```
certutil -hashfile D:\java-sdk.zip SHA256
```

The following is an example command output:

```
SHA-256 hash value of D:\java-sdk.zip
a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
CertUtil: -hashfile command executed.
```

Compare the SHA-256 value of the downloaded SDK package with that provided in the following table. If they are the same, no tampering or packet loss occurred during the package download.

Language	SHA-256 Value of the SDK Package
Java	a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
Go	caf22350181a4ecd49dc0d3f56097b10c1794792adae407140950be4ed9b6771
Python	c80b9ef282fc88d3fb16db4cb9d7525d3db71f7989782ed0b636920ea2fadb93
C#	b0e69ef60a561c54c1b86c3447ca855088a1fa2a672fbfba618aaf56b2841e8a

Language	SHA-256 Value of the SDK Package
JavaScript	c64e595651de079766e446ce2c3262013256f81683bb9434bee27bed3a4caf01
PHP	e2eba2cae72aea794edb4057ed8eb7cb82f0dbaccabf9c5539694a7a7a9f3c89
C++	c173f59d816aba53f47750cf5ffdc65cc345b1613974b3d2a545ace48787f577
C	e4f22beb7b132fe6e57c9de79f596c3ff830228cd7221b02ca96198e501c642c
Android	d6c3032801ac88cf8cbc51f64d42457174447c8d159f34a187c036913b31ea2b

----End

Obtain the **ApiGateway-cpp-sdk.zip** package. The following table shows the files decompressed from the package.

Name	Description
hasher.cpp	SDK code
hasher.h	
header.h	
RequestParams.cpp	
RequestParams.h	
signer.cpp	
signer.h	
Makefile	**Makefile** file
main.cpp	Sample code

API Calling Example

Step 1 Add the following references to **main.cpp**:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <curl/curl.h>
#include "signer.h"
```

Step 2 Generate a new signer and enter the AppKey and AppSecret.

```
Signer signer("4f5f626b-073f-402f-a1e0-e52171c6100c", "******");
```

Step 3 Generate a new **RequestParams** request, and specify the method, domain name, request URI, query strings, and request body.


```
RequestParams* request = new RequestParams("POST", "{apig-endpoint}", "/app1",
    "Action=ListUsers&Version=2010-05-08", "demo");
```

Step 4 Add a header to the request. The header contains specific parameters. Add other headers to be signed as necessary.

```
request->addHeader("x-stage", "RELEASE");
request->addHeader("name","value");
```

Step 5 Execute the following function to add the generated headers to the request variable. Then, add the **x-Authorization** header to the request. The value of the x-Authorization header is the same as that of the **Authorization** header.

```
signer.createSignature(request);
for (auto header : *request->getHeaders()) {
    if( strcmp(header.getKey().data(), "Authorization") == 0){
        request->addHeader("x-Authorization", header.getValue());
    }
}
```

Step 6 Use the curl library to access the API and view the access result.

```
static size_t
WriteMemoryCallback(void *contents, size_t size, size_t nmemb, void *userp)
{
    size_t realsize = size * nmemb;
    struct MemoryStruct *mem = (struct MemoryStruct *)userp;

    mem->memory = (char*)realloc(mem->memory, mem->size + realsize + 1);
    if (mem->memory == NULL) {
        /* out of memory! */
        printf("not enough memory (realloc returned NULL)\n");
        return 0;
    }

    memcpy(&(mem->memory[mem->size]), contents, realsize);
    mem->size += realsize;
    mem->memory[mem->size] = 0;

    return realsize;
}

//send http request using curl library
int perform_request(RequestParams* request)
{
    CURL *curl;
    CURLcode res;
    struct MemoryStruct resp_header;
    resp_header.memory = (char*)malloc(1);
    resp_header.size = 0;
    struct MemoryStruct resp_body;
    resp_body.memory = (char*)malloc(1);
    resp_body.size = 0;

    curl_global_init(CURL_GLOBAL_ALL);
    curl = curl_easy_init();

    curl_easy_setopt(curl, CURLOPT_CUSTOMREQUEST, request->getMethod().c_str());
    std::string url = "http://" + request->getHost() + request->getUri() + "?" + request->getQueryParams();
    curl_easy_setopt(curl, CURLOPT_URL, url.c_str());
    struct curl_slist *chunk = NULL;
    std::set<Header>::iterator it;
    for (auto header : *request->getHeaders()) {
        std::string headerEntry = header.getKey() + ": " + header.getValue();
        printf("%s\n", headerEntry.c_str());
        chunk = curl_slist_append(chunk, headerEntry.c_str());
    }
    printf("-------------\n");
    curl_easy_setopt(curl, CURLOPT_HTTPHEADER, chunk);
    curl_easy_setopt(curl, CURLOPT_COPYPOSTFIELDS, request->getPayload().c_str());
    curl_easy_setopt(curl, CURLOPT_NOBODY, 0L);
```



```
    curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, WriteMemoryCallback);
    curl_easy_setopt(curl, CURLOPT_HEADERDATA, (void *)&resp_header);
    curl_easy_setopt(curl, CURLOPT_WRITEDATA, (void *)&resp_body);
    //curl_easy_setopt(curl, CURLOPT_VERBOSE, 1L);
    res = curl_easy_perform(curl);
    if (res != CURLE_OK) {
        fprintf(stderr, "curl_easy_perform() failed: %s\n", curl_easy_strerror(res));
    }
    else {
        long status;
        curl_easy_getinfo(curl, CURLINFO_HTTP_CODE, &status);
        printf("status %d\n", status);
        printf(resp_header.memory);
        printf(resp_body.memory);
    }
    free(resp_header.memory);
    free(resp_body.memory);
    curl_easy_cleanup(curl);

    curl_global_cleanup();

    return 0;
}
```

Step 7 Run the **make** command to obtain a **main** executable file, execute the file, and then view the execution result.

----End

1.4.9 C

Scenarios

To use C to call an API through App authentication, obtain the C SDK, and then call the API by referring to the API calling example.

Preparing the Environment

1. You have obtained the domain name, request URL, and request method of the API to be called, and the AppKey and AppSecret of the App for calling the API. For more information, see **Preparation**.
2. Install the OpenSSL library.
   ```
   apt-get install libssl-dev
   ```
3. Install the curl library.
   ```
   apt-get install libcurl4-openssl-dev
   ```

Obtaining the SDK

Step 1 Log in to the DataArts Studio console.

Step 2 Click **DataArts DataService**.

Step 3 In the navigation pane, choose **DataArts DataService Exclusive** > **SDKs**.

Step 4 On the **SDKs** page, download the SDK package.

Step 5 Verify integrity of the SDK package. In Windows, open the CLI and run the following command to generate the SHA-256 value of the downloaded SDK package. In the command, **D:\java-sdk.zip** is an example local path and name of the SDK package. Replace it with the actual value.


```
certutil -hashfile D:\java-sdk.zip SHA256
```

The following is an example command output:

```
SHA-256 hash value of D:\java-sdk.zip
a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
CertUtil: -hashfile command executed.
```

Compare the SHA-256 value of the downloaded SDK package with that provided in the following table. If they are the same, no tampering or packet loss occurred during the package download.

Language	SHA-256 Value of the SDK Package
Java	a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
Go	caf22350181a4ecd49dc0d3f56097b10c1794792adae407140950be4ed9b6771
Python	c80b9ef282fc88d3fb16db4cb9d7525d3db71f7989782ed0b636920ea2fadb93
C#	b0e69ef60a561c54c1b86c3447ca855088a1fa2a672fbfba618aaf56b2841e8a
JavaScript	c64e595651de079766e446ce2c3262013256f81683bb9434bee27bed3a4caf01
PHP	e2eba2cae72aea794edb4057ed8eb7cb82f0dbaccabf9c5539694a7a7a9f3c89
C++	c173f59d816aba53f47750cf5ffdc65cc345b1613974b3d2a545ace48787f577
C	e4f22beb7b132fe6e57c9de79f596c3ff830228cd7221b02ca96198e501c642c
Android	d6c3032801ac88cf8cbc51f64d42457174447c8d159f34a187c036913b31ea2b

----End

Obtain the **ApiGateway-c-sdk.zip** package. The following table shows the files decompressed from the package.

Name	Description
signer_common.c	SDK code
signer_common.h	
signer.c	
signer.h	
Makefile	**Makefile** file

Name	Description
main.c	Sample code

API Calling Example

Step 1 Add the following references to **main.c**:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <curl/curl.h>
#include "signer.h"
```

Step 2 Generate a sig_params_t variable, and enter the AppKey and AppSecret.

```
sig_params_t params;
sig_params_init(&params);
sig_str_t app_key = sig_str("4f5f626b-073f-402f-a1e0-e52171c6100c");
sig_str_t app_secret = sig_str("******");
params.key = app_key;
params.secret = app_secret;
```

Step 3 Specify the method, domain name, request URI, query strings, and request body.

```
sig_str_t host = sig_str("{apig-endpoint}");
sig_str_t method = sig_str("GET");
sig_str_t uri = sig_str("/app1");
sig_str_t query_str = sig_str("a=1&b=2");
sig_str_t payload = sig_str("");
params.host = host;
params.method = method;
params.uri = uri;
params.query_str = query_str;
params.payload = payload;
```

Step 4 Add a header to the request. The header contains specific parameters. Add other headers to be signed as necessary.

```
sig_headers_add(&params.headers, "x-stage", "RELEASE");
sig_headers_add(&params.headers, "name","value");
```

Step 5 Execute the following function to add the generated headers to the request variable. Then, add the **x-Authorization** header to the request. The value of the x-Authorization header is the same as that of the **Authorization** header.

```
sig_sign(&params);
char* authorization = sig_headers_get(&params.headers, "Authorization")->value.data;
sig_headers_add(&params.headers, "x-Authorization", authorization);
```

Step 6 Use the curl library to access the API and view the access result.

```
static size_t
WriteMemoryCallback(void *contents, size_t size, size_t nmemb, void *userp)
{
    size_t realsize = size * nmemb;
    struct MemoryStruct *mem = (struct MemoryStruct *)userp;

    mem->memory = (char*)realloc(mem->memory, mem->size + realsize + 1);
    if (mem->memory == NULL) {
        /* out of memory! */
        printf("not enough memory (realloc returned NULL)\n");
        return 0;
    }

    memcpy(&(mem->memory[mem->size]), contents, realsize);
    mem->size += realsize;
```



```
    mem->memory[mem->size] = 0;

    return realsize;
}

//send http request using curl library
int perform_request(RequestParams* request)
{
    CURL *curl;
    CURLcode res;
    struct MemoryStruct resp_header;
    resp_header.memory = malloc(1);
    resp_header.size = 0;
    struct MemoryStruct resp_body;
    resp_body.memory = malloc(1);
    resp_body.size = 0;

    curl_global_init(CURL_GLOBAL_ALL);
    curl = curl_easy_init();

    curl_easy_setopt(curl, CURLOPT_CUSTOMREQUEST, params.method.data);
    char url[1024];
    sig_snprintf(url, 1024, "http://%V%V?%V", &params.host, &params.uri, &params.query_str);
    curl_easy_setopt(curl, CURLOPT_URL, url);
    struct curl_slist *chunk = NULL;
    for (int i = 0; i < params.headers.len; i++) {
        char header[1024];
        sig_snprintf(header, 1024, "%V: %V", &params.headers.data[i].name, &params.headers.data[i].value);
        printf("%s\n", header);
        chunk = curl_slist_append(chunk, header);
    }
    printf("-------------\n");
    curl_easy_setopt(curl, CURLOPT_HTTPHEADER, chunk);
    curl_easy_setopt(curl, CURLOPT_POSTFIELDS, params.payload.data);
    curl_easy_setopt(curl, CURLOPT_NOBODY, 0L);
    curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, WriteMemoryCallback);
    curl_easy_setopt(curl, CURLOPT_HEADERDATA, (void *)&resp_header);
    curl_easy_setopt(curl, CURLOPT_WRITEDATA, (void *)&resp_body);
    //curl_easy_setopt(curl, CURLOPT_VERBOSE, 1L);
    res = curl_easy_perform(curl);
    if (res != CURLE_OK) {
        fprintf(stderr, "curl_easy_perform() failed: %s\n", curl_easy_strerror(res));
    }
    else {
        long status;
        curl_easy_getinfo(curl, CURLINFO_HTTP_CODE, &status);
        printf("status %d\n", status);
        printf(resp_header.memory);
        printf(resp_body.memory);
    }
    free(resp_header.memory);
    free(resp_body.memory);
    curl_easy_cleanup(curl);

    curl_global_cleanup();

    //free signature params
    sig_params_free(&params);
    return 0;
}
```

Step 7 Run the **make** command to obtain a **main** executable file, execute the file, and then view the execution result.

----End

1.4.10 Android

Scenarios

To use Android to call an API through App authentication, obtain the Android SDK, create a project, and then call the API by referring to the API calling example.

Preparing the Environment

- You have obtained the domain name, request URL, and request method of the API to be called, and the AppKey and AppSecret of the App for calling the API. For more information, see **Preparation**.
- You have installed Android Studio. If not, download Android Studio from the **official Android Studio website** and install it.

Obtaining the SDK

Step 1 Log in to the DataArts Studio console.

Step 2 Click **DataArts DataService**.

Step 3 In the navigation pane, choose **DataArts DataService Exclusive** > **SDKs**.

Step 4 On the **SDKs** page, download the SDK package.

Step 5 Verify integrity of the SDK package. In Windows, open the CLI and run the following command to generate the SHA-256 value of the downloaded SDK package. In the command, **D:\java-sdk.zip** is an example local path and name of the SDK package. Replace it with the actual value.

```
certutil -hashfile D:\java-sdk.zip SHA256
```

The following is an example command output:

```
SHA-256 hash value of D:\java-sdk.zip
a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
CertUtil: -hashfile command executed.
```

Compare the SHA-256 value of the downloaded SDK package with that provided in the following table. If they are the same, no tampering or packet loss occurred during the package download.

Language	SHA-256 Value of the SDK Package
Java	a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
Go	caf22350181a4ecd49dc0d3f56097b10c1794792adae407140950be4ed9b6771
Python	c80b9ef282fc88d3fb16db4cb9d7525d3db71f7989782ed0b636920ea2fadb93
C#	b0e69ef60a561c54c1b86c3447ca855088a1fa2a672fbfba618aaf56b2841e8a

Language	SHA-256 Value of the SDK Package
JavaScript	c64e595651de079766e446ce2c3262013256f81683bb9434bee27bed3a4caf01
PHP	e2eba2cae72aea794edb4057ed8eb7cb82f0dbaccabf9c5539694a7a7a9f3c89
C++	c173f59d816aba53f47750cf5ffdc65cc345b1613974b3d2a545ace48787f577
C	e4f22beb7b132fe6e57c9de79f596c3ff830228cd7221b02ca96198e501c642c
Android	d6c3032801ac88cf8cbc51f64d42457174447c8d159f34a187c036913b31ea2b

----End

Obtain the **ApiGateway-android-sdk.zip** package. The following table shows the files decompressed from the package.

Name	Description
app\	Android project code
gradle\	Gradle files
build.gradle	Gradle configuration files
gradle.properties	
settings.gradle	
gradlew	Gradle Wrapper scripts
gradlew.bat	

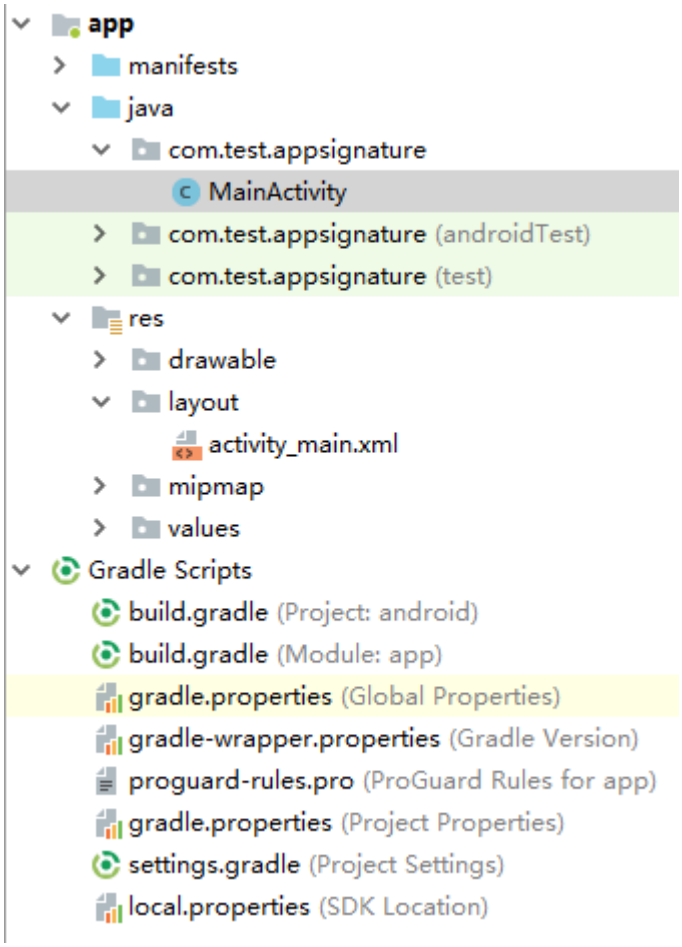
Opening a Project

Step 1 Start the Android Studio and choose **File** > **Open**.

Select the directory where the SDK is decompressed.

Step 2 View the directory structure shown in the following figure.

Figure 1-36 Project directory structure



----End

API Calling Example

Step 1 Add required JAR files to the **app/libs** directory of the Android project. The following JAR files must be included:

- java-sdk-core-x.x.x.jar
- commons-logging-1.2.jar
- joda-time-2.9.9.jar

Step 2 Add dependencies of the **okhttp** library to the **build.gradle** file.

Add **implementation 'com.squareup.okhttp3:okhttp:3.11.0'** in the **dependencies** field of the **build.gradle** file.

```
dependencies {
    ...
    ...
    implementation 'com.squareup.okhttp3:okhttp:3.11.0'
}
```

Step 3 Create a request, enter an AppKey and AppSecret, and specify the domain name, method, request URI, and body.

```
Request request = new Request();
try {
    request.setKey("4f5f626b-073f-402f-a1e0-e52171c6100c");
```



```
    request.setSecrect("******");
    request.setMethod("POST");
    request.setUrl("https://{apig-endpoint}/app1");
    request.addQueryStringParam("name", "value");
    request.addHeader("Content-Type", "text/plain");
        request.addHeader("name", "value");
    request.setBody("demo");
} catch (Exception e) {
    e.printStackTrace();
    return;
}
```

Step 4 Sign the request and add the **x-Authorization** header to the request. The value of the **x-Authorization** header is the same as that of the **Authorization** header. The **okhttp3.Request** object is then generated to access the API.

```
okhttp3.Request signedRequest = Client.signOkhttp(request);
String authorization = signedRequest.header("Authorization");
signedRequest = signedRequest.newBuilder().addHeader("x-Authorization",authorization).build();
OkHttpClient client = new OkHttpClient.Builder().build();
Response response = client.newCall(signedRequest).execute();
```

----End

1.4.11 curl

Scenarios

To use the curl command to call an API through App authentication, download the JavaScript SDK to generate the curl command, and copy the command to the CLI to call the API.

Prerequisites

You have obtained the domain name, request URL, and request method of the API to be called, and the AppKey and AppSecret of the App for calling the API. For more information, see **Preparation**.

Obtaining the SDK

Step 1 Log in to the DataArts Studio console.

Step 2 Click **DataArts DataService**.

Step 3 In the navigation pane, choose **DataArts DataService Exclusive** > **SDKs**.

Step 4 On the **SDKs** page, download the SDK package.

Step 5 Verify integrity of the SDK package. In Windows, open the CLI and run the following command to generate the SHA-256 value of the downloaded SDK package. In the command, **D:\java-sdk.zip** is an example local path and name of the SDK package. Replace it with the actual value.

```
certutil -hashfile D:\java-sdk.zip SHA256
```

The following is an example command output:

```
SHA-256 hash value of D:\java-sdk.zip
a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
CertUtil: -hashfile command executed.
```


Compare the SHA-256 value of the downloaded SDK package with that provided in the following table. If they are the same, no tampering or packet loss occurred during the package download.

Language	SHA-256 Value of the SDK Package
Java	a7c0195ebf297f29ab0065da16d7e84f14911de177e6e0c8dbadf3464d12b75f
Go	caf22350181a4ecd49dc0d3f56097b10c1794792adae407140950be4ed9b6771
Python	c80b9ef282fc88d3fb16db4cb9d7525d3db71f7989782ed0b636920ea2fadb93
C#	b0e69ef60a561c54c1b86c3447ca855088a1fa2a672fbfba618aaf56b2841e8a
JavaScript	c64e595651de079766e446ce2c3262013256f81683bb9434bee27bed3a4caf01
PHP	e2eba2cae72aea794edb4057ed8eb7cb82f0dbaccabf9c5539694a7a7a9f3c89
C++	c173f59d816aba53f47750cf5ffdc65cc345b1613974b3d2a545ace48787f577
C	e4f22beb7b132fe6e57c9de79f596c3ff830228cd7221b02ca96198e501c642c
Android	d6c3032801ac88cf8cbc51f64d42457174447c8d159f34a187c036913b31ea2b

----End

Obtain the **ApiGateway-javascript-sdk.zip** package. The following table shows the files decompressed from the package.

Name	Description
signer.js	SDK code
node_demo.js	Node.js sample code
demo.html	Browser sample code
demo_require.html	Browser sample code (loaded using **require**)
test.js	Test cases
js\hmac-sha256.js	Dependency libraries
js\moment.min.js	
js\moment-timezone-with-data.min.js	

Name	Description
licenses\license-crypto-js	Third-party library license files
licenses\license-moment	
licenses\license-moment-timezone	
licenses\license-node	

API Calling Example

Step 1 Use the JavaScript SDK to generate the curl command.

Obtain and decompress **ApiGateway-javascript-sdk.zip**. Open **demo.html** in a browser. The following figure shows the demo page.

Step 2 Specify the key, secret, method, protocol, domain name, and URL. For example:

```
Key=4f5f626b-073f-402f-a1e0-e52171c6100c
Secret=******
Method=POST
Url=https://{apig-endpoint}
```

Step 3 Specify query and header parameters in JSON format, and set the request body. The ID of the accessed API is required. You need to enter the specific ID information. The **x-api-id** parameter is used in **Headers**.

Step 4 Click **Send request** to generate a **curl** command.

```
$ curl -X POST "https://{apig-endpoint}/" -H "X-Sdk-Date: 20180530T115847Z" -H "Authorization: SDK-HMAC-SHA256 Access=071fe245-9cf6-4d75-822d-c29945a1e06a, SignedHeaders=host;x-sdk-date, Signature=9e5314bd156d517******dd3e5765fdde4" -d ""
```

Step 5 Then, add the **x-Authorization** header to the command. The value of the **x-Authorization** header is the same as that of the **Authorization** header. Copy the **curl** command to the CLI to access the API.

```
$ curl -X POST "https://{apig-endpoint}/" -H "X-Sdk-Date: 20180530T115847Z" -H "Authorization: SDK-HMAC-SHA256 Access=071fe245-9cf6-4d75-822d-c29945a1e06a, SignedHeaders=host;x-sdk-date, Signature=9e5314bd156d517******dd3e5765fdde4" -H "X-Authorization: SDK-HMAC-SHA256 Access=071fe245-9cf6-4d75-822d-c29945a1e06a, SignedHeaders=host;x-sdk-date, Signature=9e5314bd156d517******dd3e5765fdde4" -d ""
Congratulations, sdk demo is running
```

----End

1.4.12 Other Programming Languages

App Authentication Principle

1. Construct a standard request.

   Assemble the request content according to the rules of API Gateway, ensuring that the client signature is consistent with that in the backend request.

2. Create a to-be-signed string using the standard request and other related information.

3. Calculate a signature using the AK/SK and to-be-signed string.
4. Add the generated signature to an HTTP request as a header or query parameter.
5. After receiving the request, API Gateway performs **1** to **3** to calculate a signature.
6. The new signature generated in **3** is compared with the signature generated in **5**. If they are consistent, the request is processed; otherwise, the request is rejected.

NOTE

The body of a signing request in app authentication mode cannot exceed 12 MB.

Step 1: Constructing a Standard Request

To access an API through app authentication, standardize the request content, and then sign the request. The client must follow the same request specifications as API Gateway so that each HTTP request can obtain the same signing result from the frontend and backend to complete identity authentication.

The pseudocode of standard HTTP requests is as follows:

```
CanonicalRequest =
    HTTPRequestMethod + '\n' +
    CanonicalURI + '\n' +
    CanonicalQueryString + '\n' +
    CanonicalHeaders + '\n' +
    SignedHeaders + '\n' +
    HexEncode(Hash(RequestPayload))
```

The following example shows how to construct a standard request.

Original request:

```
GET https://{apig-endpoint}/app1?b=2&a=1 HTTP/1.1
Host: {apig-endpoint}
X-Sdk-Date: 20180330T123600Z
```

1. Specify an HTTP request method (**HTTPRequestMethod**) and end with a carriage return line feed (CRLF).

   Specify GET, PUT, POST, or another request method. Example of a request method:

   ```
   GET
   ```

2. Add a standard URI (**CanonicalURI**) and end with a CRLF.

   Description

   Path of the requested resource, which is the URI code of the absolute path.

   Format

   According to RFC 3986, each part of a standard URI except the redundant and relative paths must be URI-encoded. If a URI does not end with a slash (/), add a slash at its end.

   Example

   For the URI **/app1**, the standard URI code is as follows:

   ```
   GET
   /app1/
   ```


NOTE

During signature calculation, the URI must end with a slash (/). When a request is sent, the URI does not need to end with a slash (/).

3. Add a standard query string (**CanonicalQueryString**) and end with a CRLF.

   Description

   Query parameters. If no query parameters are configured, the query string is an empty string.

   Format

   Standard query strings must meet the following requirements:

   - Perform URI encoding on each parameter and value according to the following rules:
     - Do not perform URI encoding on any non-reserved characters defined in RFC 3986, including A–Z, a–z, 0–9, hyphen (-), underscore (_), period (.), and tilde (~).
     - Use **%XY** to perform percent encoding on all non-reserved characters. **X** and **Y** indicate hexadecimal characters (0–9 and A–F). For example, the space character must be encoded as **%20**, and an extended UTF-8 character must be encoded in the "%XY%ZA%BC" format.
   - Add "*URI-encoded parameter name* = *URI-encoded parameter value*" to each parameter. If no value is specified, use a null string instead. The equal sign (=) is required.

     For example, in the following string that contains two parameters, the value of parameter **parm2** is null.

     ```
     parm1=value1&parm2=
     ```
   - Sort the parameters in alphabetically ascending order. For example, a parameter starting with uppercase letter **F** precedes another parameter starting with lowercase letter **b**.
   - Construct standard query strings from the first parameter after sorting.

   Example

   The following example contains two optional parameters **a** and **b**.

   ```
   GET
   /app1/
   a=1&b=2
   ```

4. Add standard headers (**CanonicalHeaders**) and end with a CRLF.

   Description

   List of standard request headers, including all HTTP message headers in the to-be-signed request. The **X-Sdk-Date** header must be included to verify the signing time, which is in the UTC time format *YYYYMMDDTHHMMSSZ* as specified in ISO 8601. When publishing an API in a non-RELEASE environment, you need to specify an environment name.

   Format

   CanonicalHeaders consists of multiple message headers, for example, **CanonicalHeadersEntry0 + CanonicalHeadersEntry1 + ...**. Each message header (**CanonicalHeadersEntry**) is in the format of **Lowercase(HeaderName) + ':' + Trimall(HeaderValue) + '\n'**.

NOTE

- **Lowercase** is a function for converting all letters into lowercase letters.
- **Trimall** is a function for deleting the spaces before and after a value.
- The last message header carries a CRLF. Therefore, an empty line appears because the **CanonicalHeaders** field also contains a CRLF according to the specifications.

Example

```
GET
/app1/
a=1&b=2
host:{apig-endpoint}
x-sdk-date:20180330T123600Z
```

NOTICE

Standard message headers must meet the following requirements:

- All letters in a header are converted to lowercase letters, and all spaces before and after the header is deleted.
- All headers are sorted in alphabetically ascending order.

For example, the original headers are as follows:

```
Host: {apig-endpoint}\n
Content-Type: application/json;charset=utf8\n
My-header1:    a   b   c  \n
X-Sdk-Date:20180330T123600Z\n
My-Header2:    "a   b   c"  \n
```

A standard header is as follows:

```
content-type:application/json;charset=utf8\n
host:{apig-endpoint}\n
my-header1:a   b   c\n
my-header2:"a   b   c"\n
x-sdk-date:20180330T123600Z\n
```

5. Add message headers (**SignedHeaders**) for request signing, and end with a CRLF.

   Description

   List of message headers used for request signing. This step is to determine which headers are used for signing the request and which headers can be ignored during request verification. The **X-Sdk-date** header must be included.

   Format

   SignedHeaders = Lowercase(HeaderName0) + ';' + Lowercase(HeaderName1) + ";" + ...

   Letters in the message headers are converted to lowercase letters. All headers are sorted alphabetically and separated with commas.

   Lowercase is a function for converting all letters into lowercase letters.

   Example

   In the following example, two message headers **host** and **x-sdk-date** are used for signing the request.

   ```
   GET
   /app1/
   a=1&b=2
   host:{apig-endpoint}
   x-sdk-date:20180330T123600Z
   ```



```
host;x-sdk-date
```

6. Use a hash function, such as SHA-256, to create a hash value based on the body (**RequestPayload**) of the HTTP or HTTPS request.

   Description

   Request message body. The message body needs two layers of conversion (**HexEncode(Hash(*RequestPayload*))**). **Hash** is a function for generating message digest. Currently, SHA-256 is supported. **HexEncode**: the Base16 encoding function for returning a digest consisting of lowercase letters. For example, **HexEncode ("m")** returns **6d** instead of **6D**. Each byte you enter is expressed as two hexadecimal characters.

   NOTE

   If **RequestPayload** is null, the null value is used for calculating a hash value.

   Example

   For a request with the GET method and an empty body, the body (empty string) after hash processing is as follows:

   ```
   GET
   /app1/
   a=1&b=2
   host:{apig-endpoint}
   x-sdk-date:20180330T123600Z

   host;x-sdk-date
   e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
   ```

7. Perform hash processing on the standard request in the same way as that on the **RequestPayload**. After hash processing, the standard request is expressed with lowercase hexadecimal strings.

   Algorithm pseudocode:
   Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))

   Example of standard request after hash processing:

   ```
   4bd8e1afe76738a332ecff075321623fb90ebb181fe79ec3e23dcb081ef15906
   ```

Step 2: Creating a To-Be-Signed String

After a standard HTTP request is constructed and the request hash value is obtained, create a to-be-signed string by combining them with the signature algorithm and signing time.

```
StringToSign =
  Algorithm + \n +
  RequestDateTime + \n +
  HashedCanonicalRequest
```

Parameters in the pseudocode are described as follows:

- **Algorithm**

  Signature algorithm. For SHA256, the value is **SDK-HMAC-SHA256**.

- **RequestDateTime**

  Request timestamp, which is the same as **X-Sdk-Date** in the request header. The format is *YYYYMMDDTHHMMSSZ*.

- **HashedCanonicalRequest**

  Standard request generated after hash processing.

In this example, the following to-be-signed string is obtained:

```
SDK-HMAC-SHA256
20180330T123600Z
4bd8e1afe76738a332ecff075321623fb90ebb181fe79ec3e23dcb081ef15906
```

Step 3: Calculating the Signature

Use the AppSecret and created character string as the input of the encryption hash function, and convert the calculated binary signature into a hexadecimal expression.

The pseudocode is as follows:

```
signature = HexEncode(HMAC(APP secret, string to sign))
```

HMAC indicates hash calculation, and **HexEncode** indicates hexadecimal conversion. **Table 1-3** describes the parameters in the pseudocode.

Table 1-3 Parameter description

Parameter	Description
AppSecret	Signature key.
To-be-signed string	Character string to be signed.

Assuming that the AppSecret is **12345678-1234-1234-1234-123456781234**, a signature similar to the following will be calculated:

```
cb978df7c06ac242bab1d1b39d697ef7df4806664a6e09d5f5308a6b25043ea2
```

Step 4: Adding the Signature to the Request Header

Add the signature to the HTTP Authorization header. The Authorization header is used for identity authentication and not included in the signed headers.

The pseudocode is as follows:

```
Authorization header creation pseudocode:
Authorization: algorithm Access=APP key, SignedHeaders=SignedHeaders, Signature=signature
```

There is no comma before the algorithm and **Access**. **SignedHeaders** and **Signature** must be separated with commas.

The signed headers are as follows:

```
Authorization: SDK-HMAC-SHA256 Access=071fe245-9cf6-4d75-822d-c29945a1e06a, SignedHeaders=host;x-sdk-date, Signature=cb978df7c06ac242bab1d1b39d697ef7df4806664a6e09d5f5308a6b25043ea2
```

After obtaining the signed message headers, add them to the original HTTP request header with the **Authorization** and **x-Authorization** parameters. The request is sent to API Gateway for identity authentication. If the identity authentication is successful, the request is sent to the corresponding backend service for processing.

