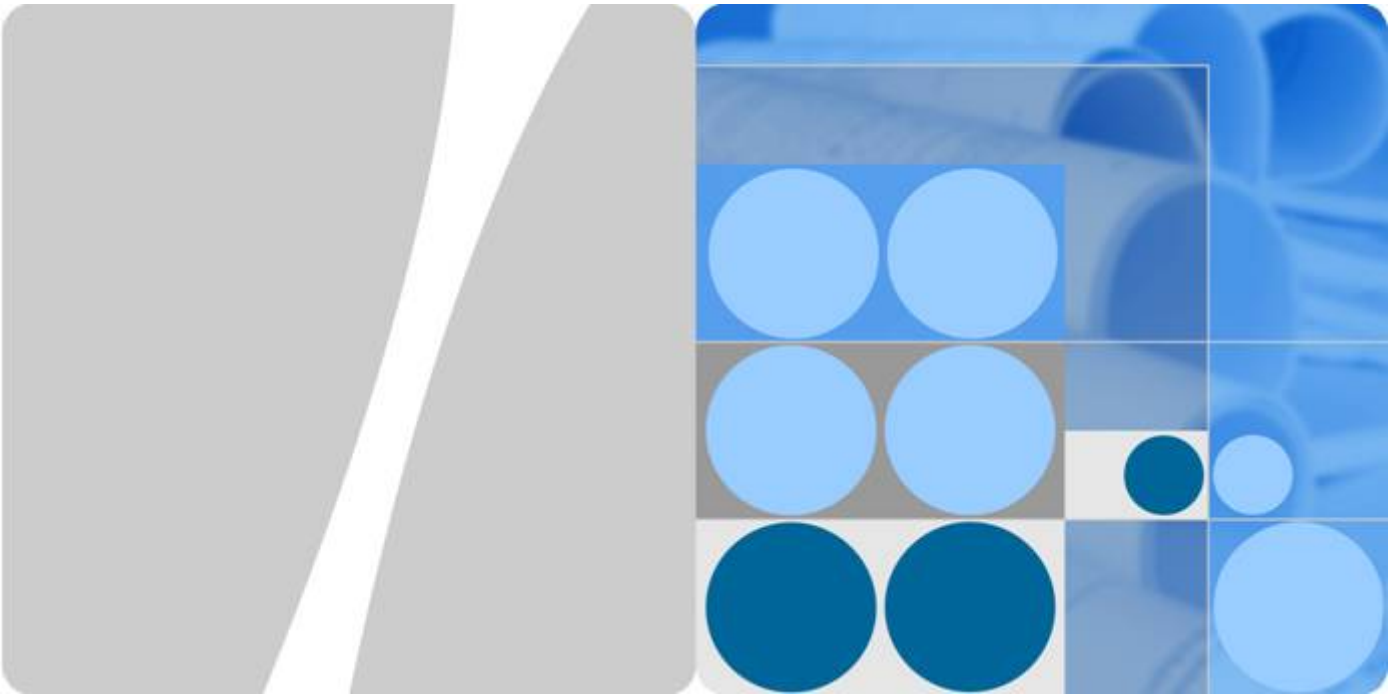


数据治理中心

2.10.1

SDK 参考

发布日期　2024-04-30

目 录

1 数据服务 SDK 参考

1.1 概述

本文档指导API调用者通过数据服务SDK代码调用数据API，当前数据服务SDK代码仅支持调用API场景。

数据服务 SDK 介绍

数据服务SDK是基于DataArts Studio数据服务创建的数据API封装的SDK包。通过调用此SDK包提供的代码样例，即可进行数据服务中数据API的调用，帮助开发者简单、快速地通过数据API获取到开放数据。

数据服务 SDK 使用场景

数据API是否必须通过数据服务SDK代码才能调用，与数据API的认证方式有关。当通过数据服务创建数据API使用推荐的APP认证方式时，必须通过SDK方式进行数据API调用；当使用其他认证方式时，可以通过API调用工具调用，也可以通过SDK方式进行数据API调用。

- “APP认证方式”：API调用者通过APP认证方式调用API。Appkey & Appsecret安全级别高，推荐使用。

  使用APP认证时，需要通过SDK访问，其中SDK访问提供了基于Java、Go、Python、JavaScript、C#、PHP、C++、C、Android等多种语言的SDK包。各语言调用API示例请参考《 数据治理中心 2.10.1 SDK参考 》的“使用APP认证调用API”章节。

- “IAM认证”：需要借助IAM服务进行安全认证。IAM认证只允许云用户访问，安全级别中等。

  使用IAM认证时，需要通过调用IAM服务的获取用户Token接口获取Token，然后通过在请求消息头中添加“X-Auth-Token”参数并将所获取的Token作为参数值，再通过API调用工具或SDK方式调用已发布的API。

- “无认证”：不需要认证。安全级别低，所有用户均可访问，建议仅在测试接口时使用，不推荐正式使用。若调用方为不可信任用户，则存在数据库安全风险（如数据泄露、数据库高并发访问导致宕机、SQL注入等风险）。

  使用无认证方式时，无需鉴权认证信息。通过API调用工具或SDK方式，直接使用已发布API域名并填写入参即可进行调用。

1.2 SDK 使用前准备

步骤1 下载SDK，并导入对应SDK到本地开发工具。

1. 登录DataArts Studio控制台。
2. 单击“数据服务”模块。
3. 单击左侧菜单“专享版 > SDK”。
4. 单击SDK使用引导区域里对应语言的SDK，下载SDK包到本地。
5. 进行SDK包完整性校验。Windows操作系统下，打开本地命令提示符框，输入如下命令，在本地生成已下载SDK包的SHA256值，其中，“D:\java-sdk.zip”为SDK包的本地存放路径和SDK包名，请根据实际情况修改。

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

   命令执行结果示例，如下所示：

   ```
   SHA256 的 D:\java-sdk.zip 哈希:
   becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
   CertUtil: -hashfile 命令成功完成。
   becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
   ```

   对比所下载SDK包的SHA256值和下表中对应语言SDK包的SHA256值。如果一致，则表示下载过程不存在篡改和丢包。

不同语言SDK包	SHA256值
Java	becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
Go	bcf8cf19a21226e247195f2e584c8414da39b8d05840fb02948e1375d9bbb7e6
Python	c3da3b5814f828d6217963e856563d558d938b3da28993a8a13c8a7ebff5b95d
C#	a880b47e63ab35bfe216592e340a8135b866aef8f756ef7738fff3287885f33a
JavaScript	53261387f5fcf46e61d0bef5e890bea97952717f327c356412c3128389e848d6
PHP	29bf711144e77a4adaea1257cd6dedd2220e57b729a8fd000c51e68ccb42ad4b
C++	f604c6386c62cccb7c358007778037d5b15480987dc2860eef1b7bad37cb21d7
C	7086012c2d0569d5938830926b19fbea0d46682a983e04e52924978e8720c2f8
Android	89962b186707828b06b0c9f50c010b2f4cefd6a8e7ca9bdefb616bbbf6e739c8

步骤2 完成请求消息入参准备。

表 1-1 参数

参数类型	参数说明	使用示例
path参数	路径参数，作为url的一部分，直接替换url中{}标识的参数	参数: param = xxx 原url: http://域名/p1/{param}/p2 实际url: http://域名/p1/xxx/p2
query参数	查询参数，作为url的补充部分	参数: param = xxx 参数2: param2 = xxx2 示例1： 调用方法添加query参数（以各语言SDK为准） 例: request.addQueryStringParam("param","xxx"); 示例2： 直接拼接到url后面，以?开头，多个参数以&连接 原url: http://域名/p1 实际url: http://域名/p1?param=xxx&param2=xxx2
header参数	请求头参数，作为请求头的一部分，参数名不区分大小写	参数: param = xxx 调用方法添加header头参数/构造请求时添加等（以各语言SDK为准） 例：request.addHeader("param"," xxx");
body参数	请求体参数，SDK中输入内容为json字符串（老版本不支持）	"{}"

步骤3 修改SDK，在请求签名后，获取请求头中的签名参数Authorization。并新增参数x-Authorization，值与Authorization相同。获取与新增方式，详见各语言调用文档**认证前准备**。

----结束

1.3 SDK 调用常见错误码/错误信息

表 1-2 常见错误码/错误信息

错误码	错误信息	错误原因	解决方案
DLM.0	null	表示API调用成功。	表示调用成功，无需处理。

错误码	错误信息	错误原因	解决方案
APIG.0101	The API does not exist or has not been published in the environment	1. API未发布 2. url错误	1. 发布API 2. 确认请求的url和实际url是否相同
APIG.0106	Orchestration error: Invalid header parameter: x-Authorization, required	SDK未添加x-Authorization	**SDK使用前准备步骤3**
APIG.0106	Orchestration error: Invalid ___ parameter: ___, required	未传指定参数	调用时上传此参数
APIG.0201	Backend timeout	后端超时（API网关请求维持50秒后未收到返回结果，会返回此错误信息）	请先确认数据服务访问日志，如果访问日志中有数据（数据略有延迟），则说明数据源取数时间过长，请尽量优化取数sql逻辑。 如果访问日志中无数据，请确认（专享版：数据服务集群）是否运行中。
APIG.0303	Incorrect app authentication information: app not found	应用不存在	确认请求的key和secret是否准确
APIG.0304	The app is not authorized to access the API	应用无权访问当前API	1. 确认API已授权给应用 2. 确认请求的key和secret是否准确
APIG.0308	The throttling threshold has been reached: policy domain over ratelimit, limit:1000, time:1 day	域名的请求次数达到了给定的上限：1天1000次	1. 建议：去API网关，为分组绑定域名 2. 临时规避：切换分组。域名以分组为单位，每个分组限制独立计算

错误码	错误信息	错误原因	解决方案
DLM.4018	Api is not exist	API不存在	20200630版本前发布的API: 请确认调用时参数x-api-id的值，是否正确（该值为所访问API的ID，请向此API的提供者获取） 20200630版本后发布的API: 1. 确认请求的url和实际url是否相同 2. 若为专享版刚发布的API，请稍作等待，API下发至集群存在短暂的延时 其他（数据同步异常）: 1. 停用/下线对应API，然后恢复/发布。 2. 尝试重启集群（逐个节点重启可避免业务影响）
DLM.4094	Call api failed.	调用API失败	1. 确认调用所执行的SQL正确，可以正常使用（所执行SQL详见访问日志，仅API方可见） 2. CDM中代理异常，错误原因详见返回的DLG错误信息。 3. 调用超时，若为DWS数据库，建议API采用自定义分页。 4. 调用超时，请优化查询语句，确保数据库中能短时间内完成执行。

错误码	错误信息	错误原因	解决方案
DLM.4211	Token invalid	token校验不通过	1. 确认token是否正确 2. 确认token所属租户，是否已授权或已位于白名单中
DLM.4312	Missing parameters: ___	缺少指定参数	调用时上传此参数
400	App does not have permission to access API.	应用无权访问当前API	1. 确认API已授权给应用 2. 确认请求的key和secret是否准确 3. 确认API和APP的授权关系仍在有效期内
401	Authorization not found.	签名信息未找到	1. 应用认证：**SDK使用前准备步骤3**。 2. 发布到网关的专享版IAM认证：IAM认证的API发布到网关后，不支持直接访问集群的token认证形式。
401	Authorization format incorrect.	签名格式错误	建议使用SDK生成签名
401	Signing key not found.	签名密钥未找到	确认请求的key和secret是否准确
401	Signed header ___ not found.	签名头未找到	请确认用于签名的header头参数在调用时上传了
401	Header x-sdk-date not found.	签名头x-sdk-date未找到	此参数为签名时自动生成，若通过其他方式调用，请将SDK签名后的此参数在调用时也进行上传

错误码	错误信息	错误原因	解决方案
401	Signature expired.	签名过期	1. 签名具有一定的有效期，当前签名已过期，请重新生成签名 2. 请确认本地时间和实际时间是否一致 3. 如果本地时间是准确的，请联系相关人员确认集群节点时间，可能节点时间存在异常
401	Verify authroization failed.	签名校验失败	请确认进行签名的所有参数，均已上传且和签名时相同，包括且不限于url、path、header、query、body等。 补充: 1. 如果自行对接了第三方网关等，请求地址和数据服务展示的地址不一致，需要在请求头中补充参数 x-forwarded-host，值为签名时使用的请求地址 2. 如果使用get请求，不要定义body体
DLG.0902	Fail to call the agent. For details about No matching constant for [-1], see the CDM logs.	CDM上的代理拒绝服务 1. SQL执行时间过长 2. CDM资源不足了	1. 确认SQL执行时长，如果时间过长，建议优化SQL（默认分页的话则建议使用自定义分页） 2. 如果SQL执行时间较短，当前没有其他服务正在作业的话，重启CDM

错误码	错误信息	错误原因	解决方案
DAYU.1088	Failed to process the request sent by the agent.	CDM无响应	1. 尝试重启CDM 2. 可能是CDM升级引起，考虑新买一个CDM

1.4 使用 APP 认证调用 API

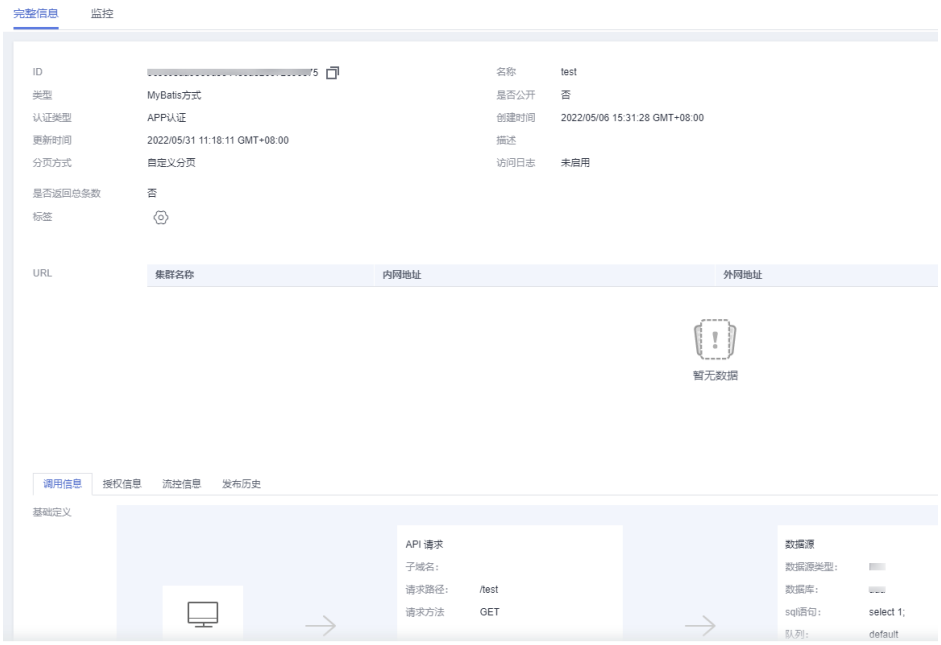
1.4.1 认证前准备

通过SDK调用API前，需要获取如下认证信息：

- 访问服务前，首先需要得到API的ID、请求URL和请求方法

  在数据服务的“API目录”页面，单击API名称，在“完整信息”页面查看API的ID、请求URL和请求方法。

  图 1-1 API 基础定义



- 对于APP认证的API，您必须提供有效的AppKey、AppSecret才能够生成认证签名。

  在“应用管理”中生成一个APP，并将APP绑定到API，就可以使用APP对应的AppKey和AppSecert访问该API。可在应用详细信息中查看AppKey和AppSecret。

  图 1-2 查看 AppKey 和 AppSecret

说明

- AppKey：APP访问密钥ID。与私有访问密钥关联的唯一标识符；访问密钥ID和私有访问密钥一起使用，对请求进行加密签名。
- AppSecret：与访问密钥ID结合使用的密钥，对请求进行加密签名，可标识发送方，并防止请求被修改。

- 发送API请求时，需要将当前时间置于HTTP的X-Sdk-Date头，将签名信息置于Authorization头。签名只在一个有限的时间内是有效的，超时即无效。

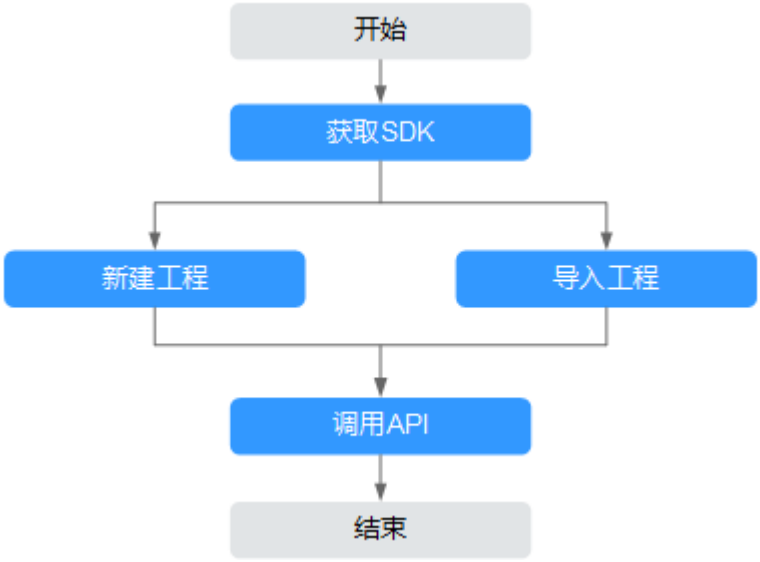
1.4.2 Java

操作场景

使用Java语言调用APP认证的API时，您需要先获取SDK，然后新建工程或导入工程，最后参考调用API示例调用API。

本章节以Eclipse 4.5.2版本为例介绍。

图 1-3 调用流程



前提条件

- 已获取API的域名、ID、请求url、请求方法、AppKey和AppSecret等信息，具体参见**认证前准备**。
- 已安装Eclipse 3.6.0或以上版本，如果未安装，请至**Eclipse官方网站**下载。
- 已安装Java Development Kit 1.8.111或以上版本，如果未安装，请至**Oracle官方下载页面**下载。

获取 SDK

步骤1 登录DataArts Studio控制台。

步骤2 单击“数据服务”模块。

步骤3 单击左侧菜单“专享版 > SDK”。

步骤4 单击SDK使用引导区域里对应语言的SDK，下载SDK包到本地。

步骤5 进行SDK包完整性校验。Windows操作系统下，打开本地命令提示符框，输入如下命令，在本地生成已下载SDK包的SHA256值，其中，“D:\java-sdk.zip”为SDK包的本地存放路径和SDK包名，请根据实际情况修改。

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

命令执行结果示例，如下所示：

```
SHA256 的 D:\java-sdk.zip 哈希:
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
CertUtil: -hashfile 命令成功完成。
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
```

对比所下载SDK包的SHA256值和下表中对应语言SDK包的SHA256值。如果一致，则表示下载过程不存在篡改和丢包。

不同语言SDK包	SHA256值
Java	becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
Go	bcf8cf19a21226e247195f2e584c8414da39b8d05840fb02948e1375d9bbb7e6
Python	c3da3b5814f828d6217963e856563d558d938b3da28993a8a13c8a7ebff5b95d
C#	a880b47e63ab35bfe216592e340a8135b866aef8f756ef7738fff3287885f33a
JavaScript	53261387f5fcf46e61d0bef5e890bea97952717f327c356412c3128389e848d6
PHP	29bf711144e77a4adaea1257cd6dedd2220e57b729a8fd000c51e68ccb42ad4b
C++	f604c6386c62cccb7c358007778037d5b15480987dc2860eef1b7bad37cb21d7
C	7086012c2d0569d5938830926b19fbea0d46682a983e04e52924978e8720c2f8
Android	89962b186707828b06b0c9f50c010b2f4cefd6a8e7ca9bdefb616bbbf6e739c8

----结束

获取“ApiGateway-java-sdk.zip”压缩包，解压后目录结构如下：

名称	说明
libs\	SDK依赖库
libs\java-sdk-core-*x.x.x*.jar	SDK包

名称	说明
src\com\apig\sdk\demo\Main.java	使用SDK签名请求示例代码
src\com\apig\sdk\demo\OkHttpDemo.java	
src\com\apig\sdk\demo\LargeFileUploadDemo.java	
src\com\apig\sdk\demo\WebSocketDemo.java	
.classpath	Java工程配置文件
.project	

导入工程

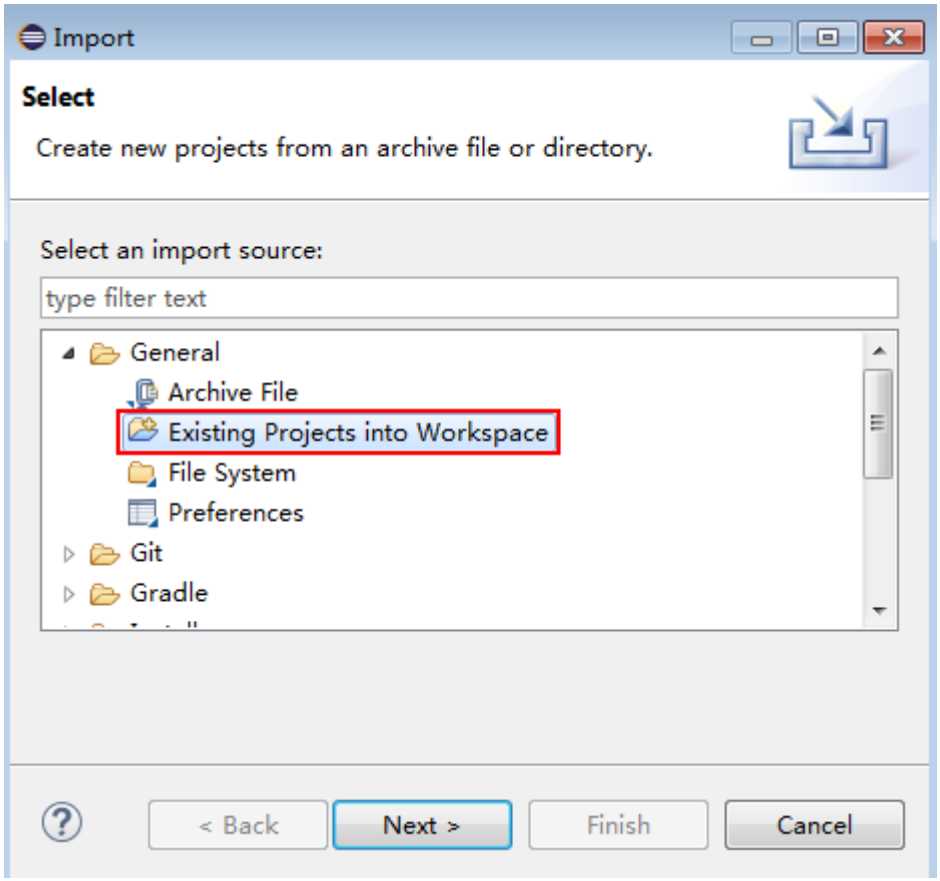
步骤1 打开Eclipse，在菜单栏选择“File > Import”。

弹出“Import”对话框。

步骤2 选择“General > Existing Projects into Workspace”，单击“Next”。

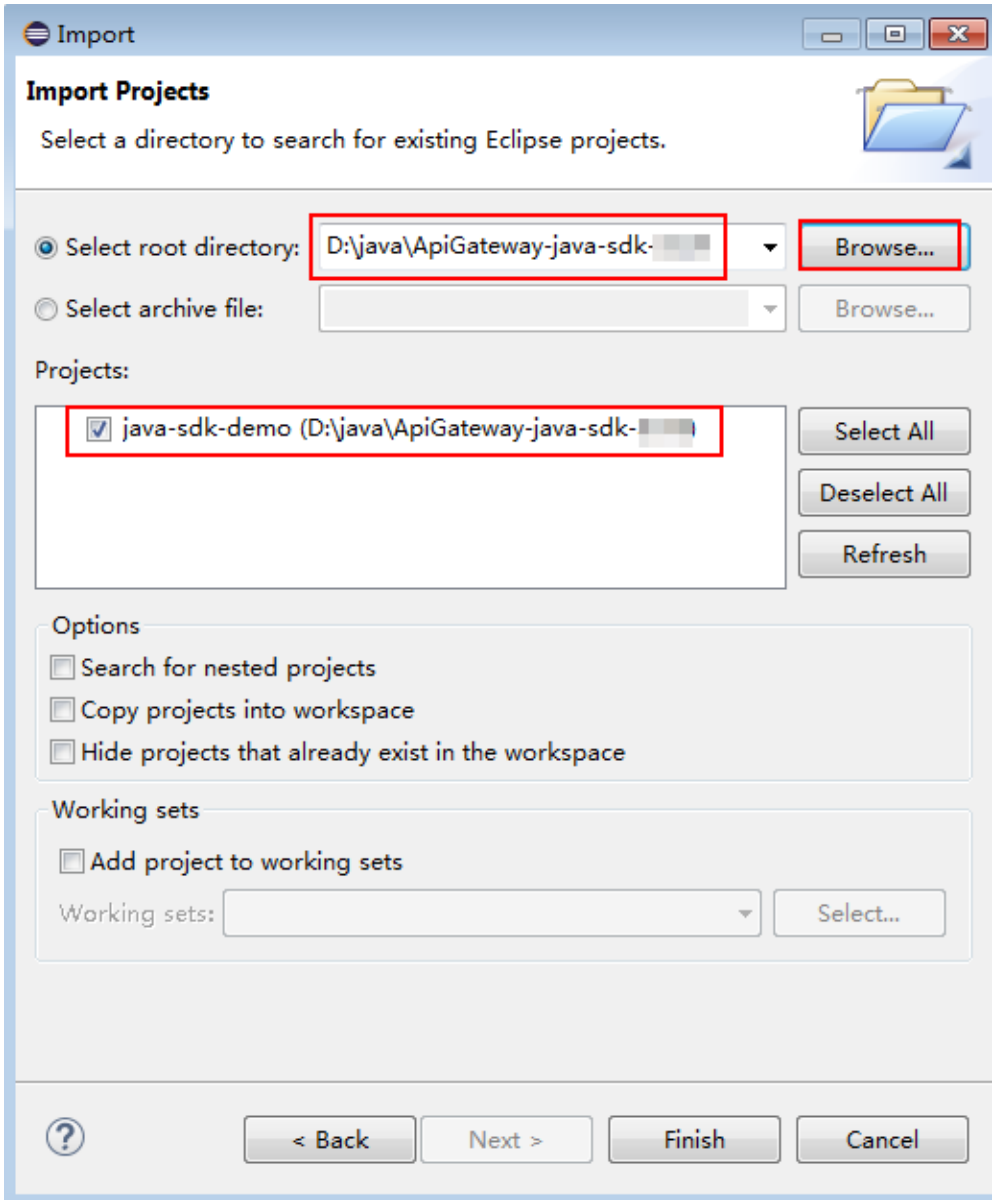
弹出“Import Projects”对话框。

图 1-4 Import



步骤3 单击“Browse”，在弹出的对话框中选择解压后的SDK路径。

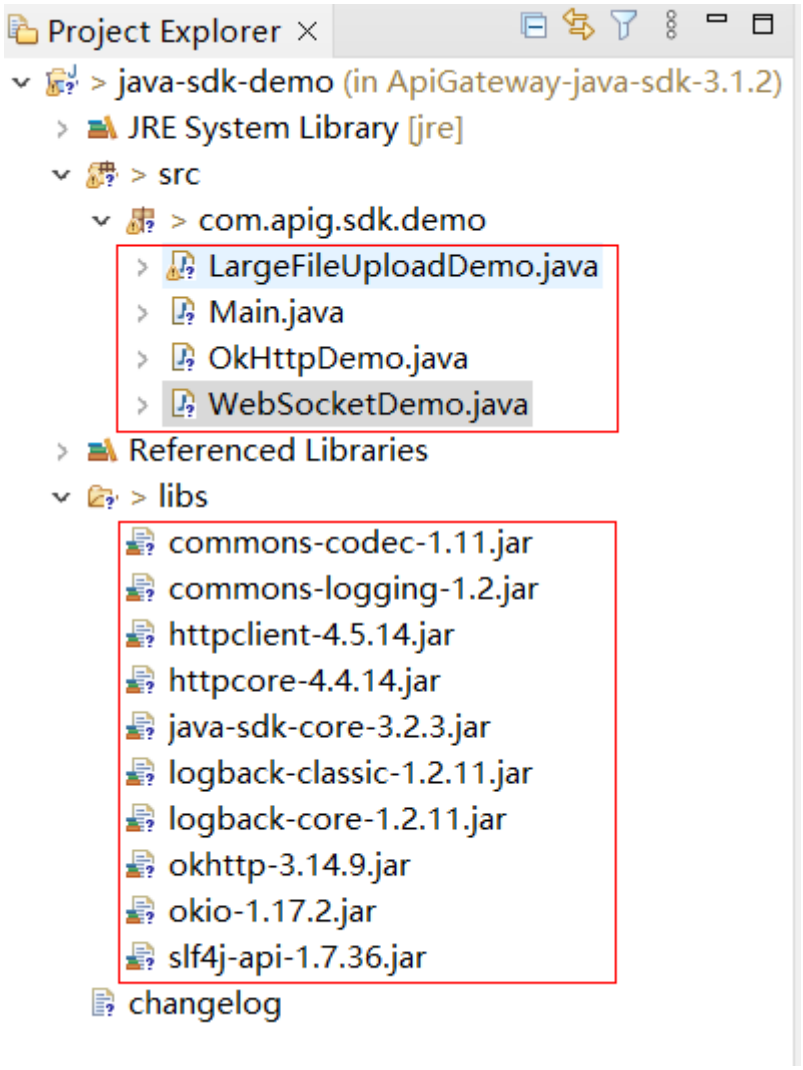
图 1-5 选择 demo 工程



步骤4 单击“Finish”，完成工程导入。

最终工程目录结构如下：

图 1-6 导入工程的目录结构



“Main.java”为示例代码，请根据实际情况修改参数后使用。具体代码说明请参考**调用API示例**。

----结束

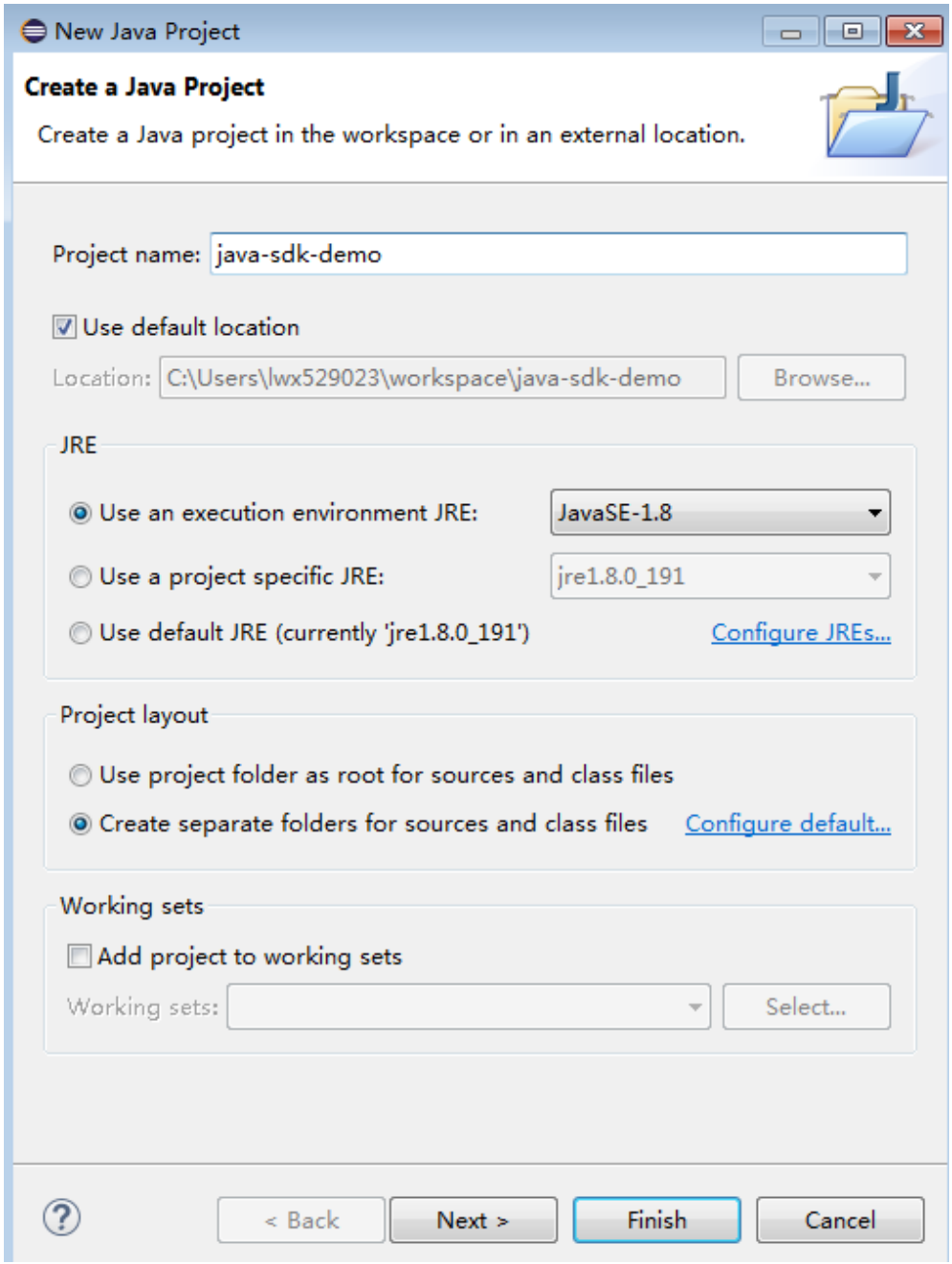
新建工程

步骤1 打开Eclipse，在菜单栏选择“File > New > Java Project”。

弹出“New Java Project”对话框。

步骤2 自定义“Project name”，以“java-sdk-demo”为例，其他参数保持默认，单击“Finish”。

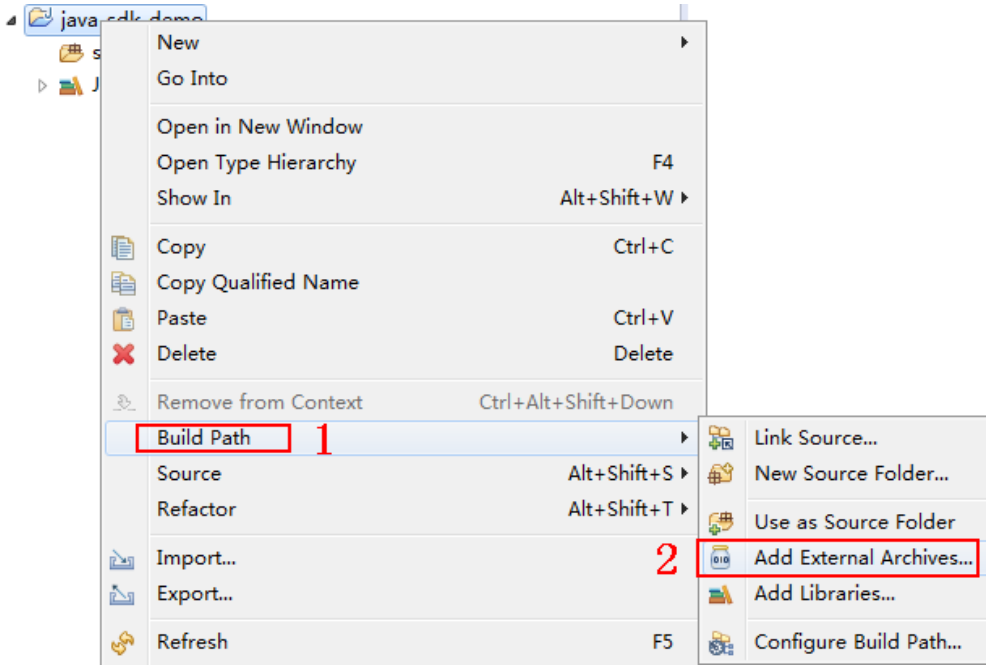
图 1-7 新建工程



步骤3 导入API Gateway Java SDK的“jar”文件。

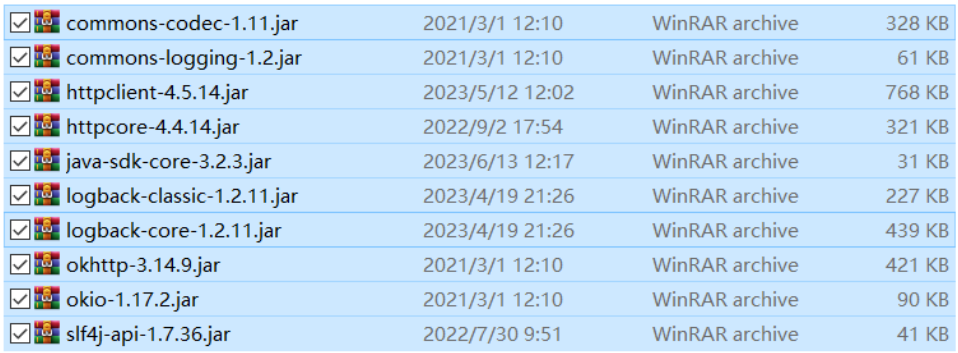
1. 选择“java-sdk-demo”，单击鼠标右键，选择“Build Path > Add External Archives”。

图 1-8 导入 jar 文件



2. 选择SDK中“\libs”目录下所有以“jar”结尾的文件，单击“打开”。

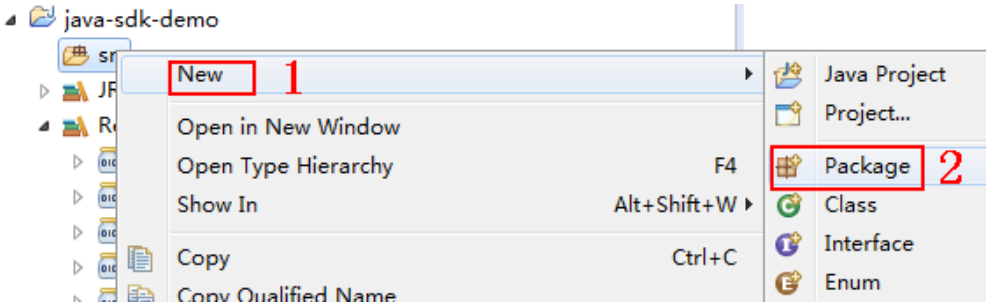
图 1-9 选择 jar 文件



步骤4 新建“Package”及“Main”文件。

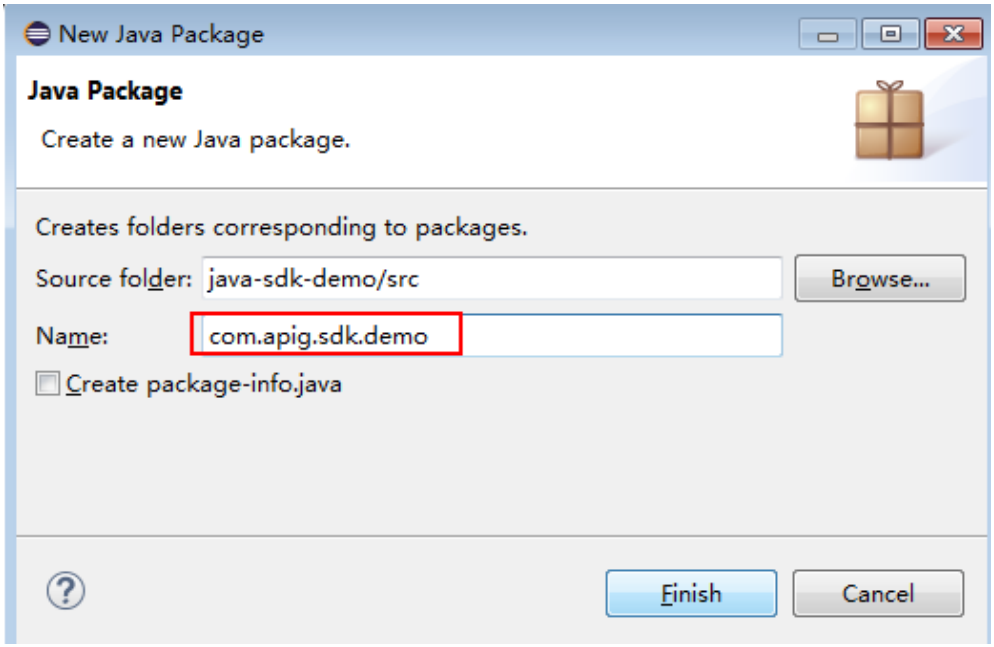
1. 选择“src”，单击鼠标右键，选择“New > Package”。

图 1-10 新建 Package



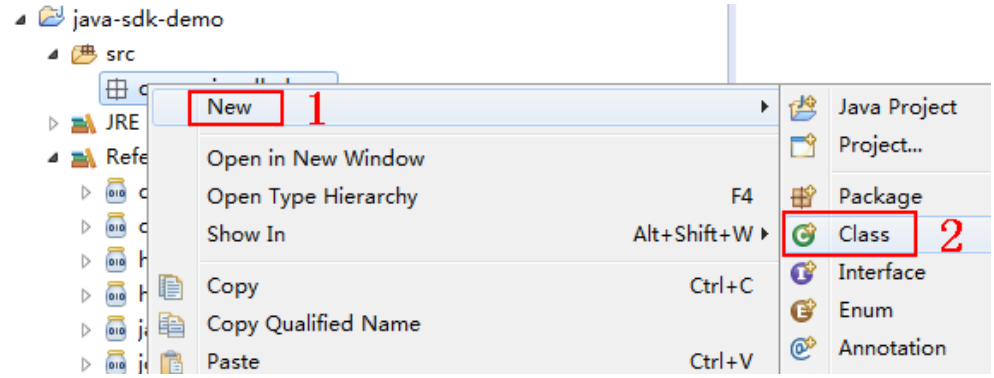
2. 在“Name”中输入“com.apig.sdk.demo”。

图 1-11 设置 Package 的名称



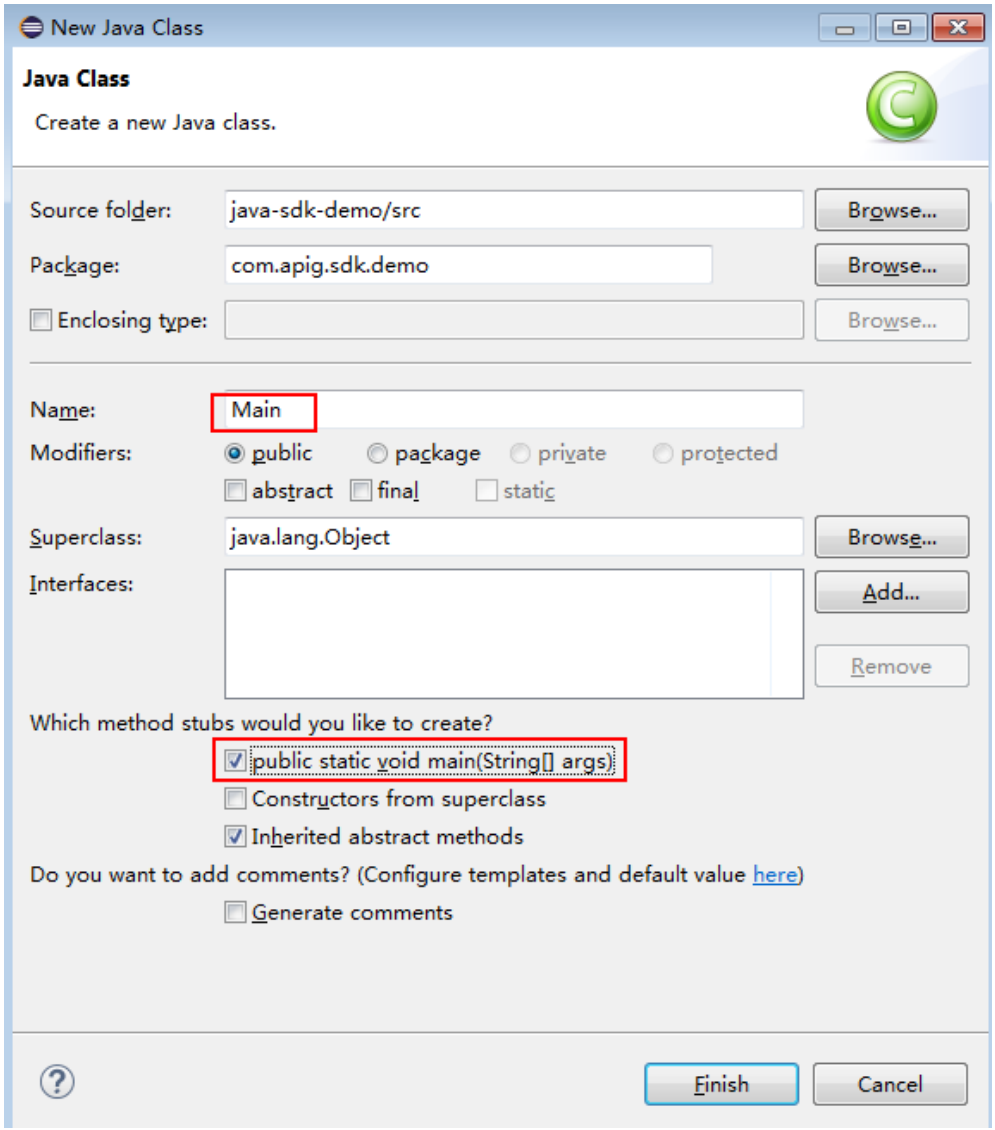
3. 单击“Finish”。
   完成“Package”的创建。
4. 选择“com.apig.sdk.demo”，单击鼠标右键，选择“New > Class”。

图 1-12 新建 Class



5. 在“Name”中输入“Main”，勾选“public static void main(String[] args)”。

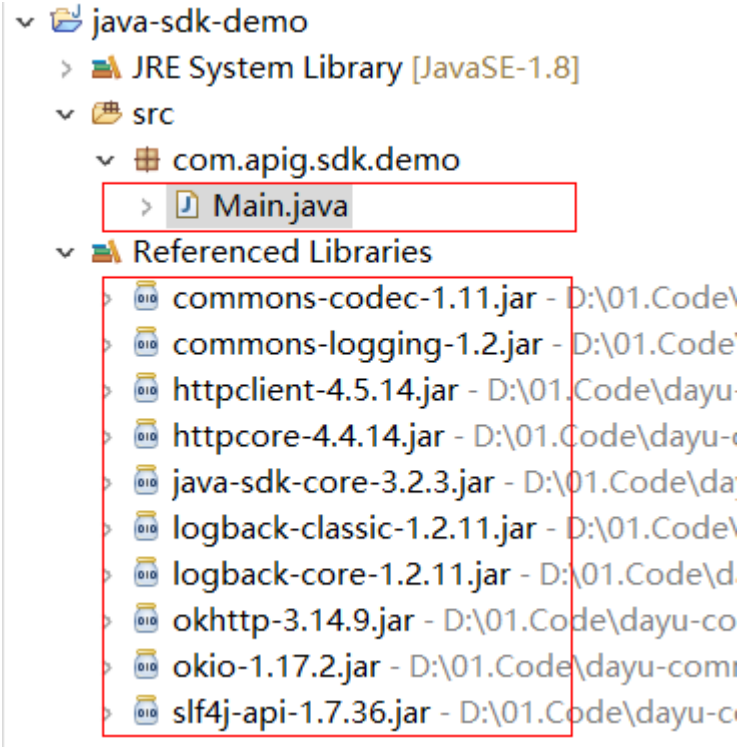
图 1-13 设置 Class 的配置



6. 单击“Finish”。
   完成“Main”文件的创建。

步骤5 完成工程创建后，最终目录结构如下。

图 1-14 新建工程的目录结构



“Main.java”无法直接使用，请根据实际情况参考**调用API示例**输入所需代码。

----结束

调用 API 示例

说明

- 示例演示如何访问发布的API。
- 您需要在数据服务控制台自行创建和发布一个API。创建及发布API的步骤请参见《数据治理中心(DataArts Studio) 用户指南》的“创建API和发布API”章节。
- 示例API的后端为打桩的HTTP服务，此后端返回一个“200”响应码及“Congratulations, sdk demo is running”消息体。

步骤1 在“Main.java”中加入以下引用。

```
import com.cloud.apigateway.sdk.utils.Client;
import com.cloud.apigateway.sdk.utils.Request;
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.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
```

步骤2 创建request，过程中需要用到如下参数。

- AppKey：通过**认证前准备**获取。认证用的ak和sk编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全，本示例从环境变量中获取。

- AppSecret：通过**认证前准备**获取。认证用的ak和sk编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全，本示例从环境变量中获取。
- Method：请求的方法名。根据API实际情况填写，示例代码使用“POST”作为样例。
- url：请求的url，不包含QueryString及fragment部分。域名部分请使用API所在的分组绑定的您自己的独立域名。示例代码使用“http://serviceEndpoint/java-sdk”作为样例。
- queryString: url携带参数的部分，根据API实际情况填写。支持的字符集为[0-9a-zA-Z./;[]\-=~#%^&_+: ']。示例代码使用“name=value”作为样例。
- header：请求的头域。根据API实际情况填写，不支持中文和下划线。示例代码使用“Content-Type:text/plain”作为样例。
- body：请求的正文。根据API实际情况填写，示例代码使用“demo”作为样例。

样例代码如下：

```
    Request request = new Request();
    try
    {
        // 认证用的ak和sk编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
        // 本示例以ak和sk保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量SDK_AK和SDK_SK。
        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");

        request.setKey(ak);
        request.setSecret(sk);

        request.setMethod("POST");
        request.setUrl("http://serviceEndpoint/java-sdk");
         //url地址在创建API分组时得到
        request.addQueryStringParam("name", "value");
        request.addHeader("Content-Type", "text/plain");
        //request.addHeader("x-stage", "publish_env_name"); //如果API发布到非RELEASE环境，需要增加自定义的环境名称
        request.setBody("demo");
    } catch (Exception e)
    {
        e.printStackTrace();
        return;
    }
```

步骤3 对请求进行签名、新增x-Authorization头、访问API并打印结果：

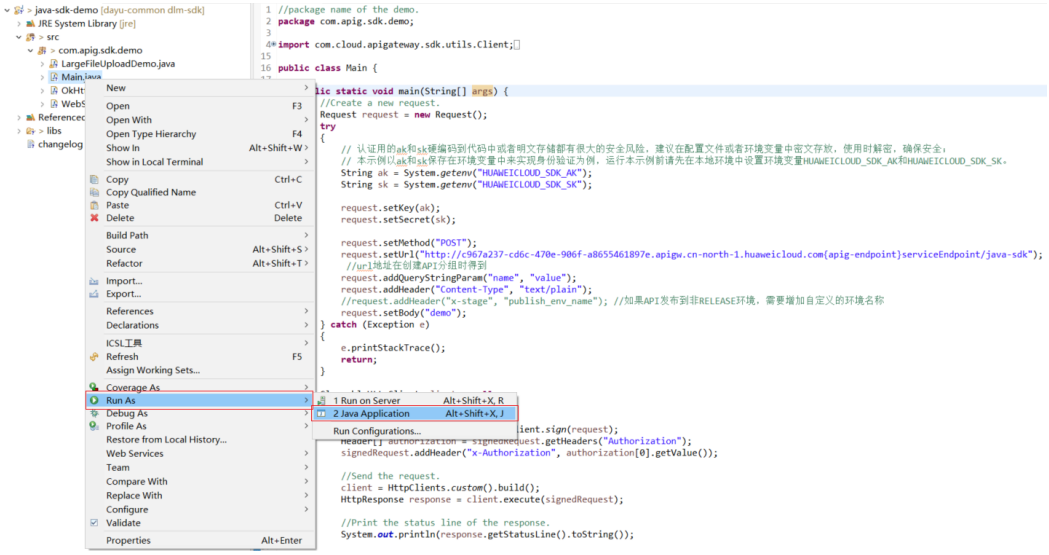
样例代码如下：

```
    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();
        }
    }
```

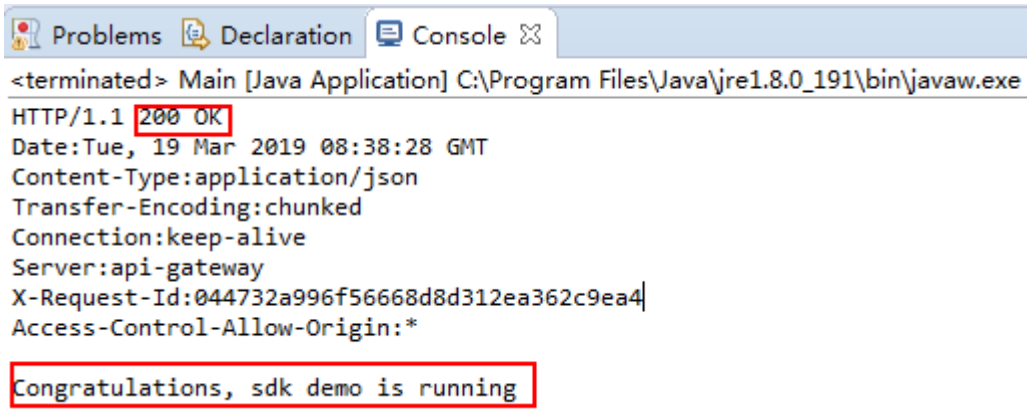
步骤4 选择“Main.java”，单击鼠标右键，选择“Run As > Java Application”，运行工程测试代码。

图 1-15 运行工程测试代码



步骤5 在“Console”页签，查看运行结果。

图 1-16 调用成功后的返回信息



----结束

1.4.3 Go

操作场景

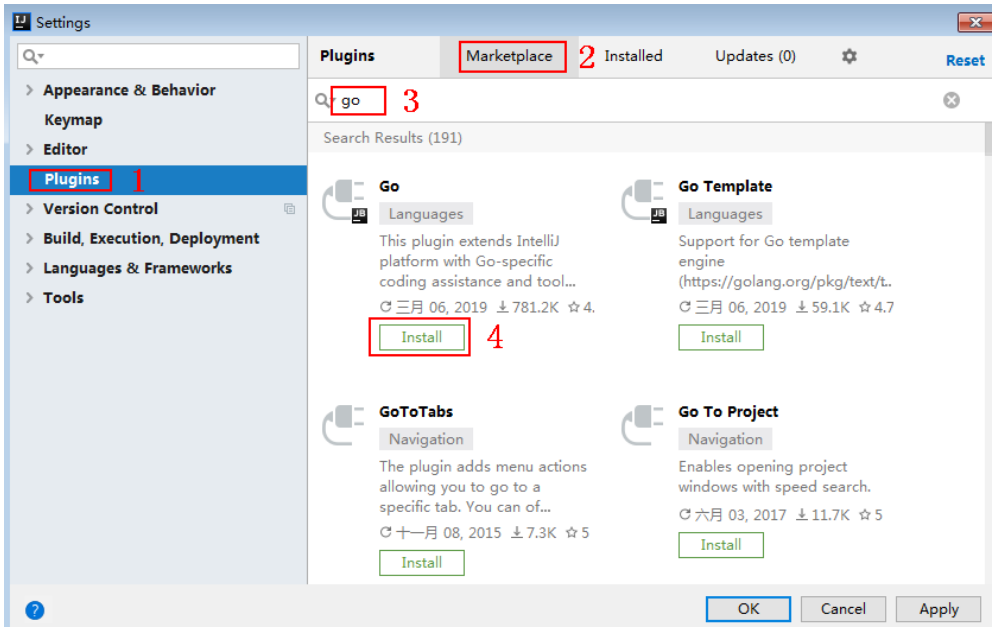
使用Go语言调用APP认证的API时，您需要先获取SDK，然后新建工程，最后参考调用API示例调用API。

本章节以IntelliJ IDEA 2018.3.5版本为例介绍。

前提条件

- 已获取API的域名、ID、请求url、请求方法、AppKey和AppSecret等信息，具体参见**认证前准备**。
- 获取并安装Go安装包，如果未安装，请至**Go官方网站**下载。
- 获取并安装IntelliJ IDEA，如果未安装，请至**IntelliJ IDEA官方网站**下载。
- 已在IntelliJ IDEA中安装Go插件，如果未安装，请按照**图1-17**所示安装。

图 1-17 安装 Go 插件



获取 SDK

步骤1 登录DataArts Studio控制台。

步骤2 单击“数据服务”模块。

步骤3 单击左侧菜单“专享版 > SDK”。

步骤4 单击SDK使用引导区域里对应语言的SDK，下载SDK包到本地。

步骤5 进行SDK包完整性校验。Windows操作系统下，打开本地命令提示符框，输入如下命令，在本地生成已下载SDK包的SHA256值，其中，“D:\java-sdk.zip”为SDK包的本地存放路径和SDK包名，请根据实际情况修改。

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

命令执行结果示例，如下所示：

```
SHA256 的 D:\java-sdk.zip 哈希:
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
CertUtil: -hashfile 命令成功完成。
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
```

对比所下载SDK包的SHA256值和下表中对应语言SDK包的SHA256值。如果一致，则表示下载过程不存在篡改和丢包。

不同语言SDK包	SHA256值
Java	becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
Go	bcf8cf19a21226e247195f2e584c8414da39b8d05840fb02948e1375d9bbb7e6
Python	c3da3b5814f828d6217963e856563d558d938b3da28993a8a13c8a7ebff5b95d

不同语言SDK包	SHA256值
C#	a880b47e63ab35bfe216592e340a8135b866aef8f756ef7738fff3287885f33a
JavaScript	53261387f5fcf46e61d0bef5e890bea97952717f327c356412c3128389e848d6
PHP	29bf711144e77a4adaea1257cd6dedd2220e57b729a8fd000c51e68ccb42ad4b
C++	f604c6386c62cccb7c358007778037d5b15480987dc2860eef1b7bad37cb21d7
C	7086012c2d0569d5938830926b19fbea0d46682a983e04e52924978e8720c2f8
Android	89962b186707828b06b0c9f50c010b2f4cefd6a8e7ca9bdefb616bbbf6e739c8

----结束

获取“ApiGateway-go-sdk.zip”压缩包，解压后目录结构如下：

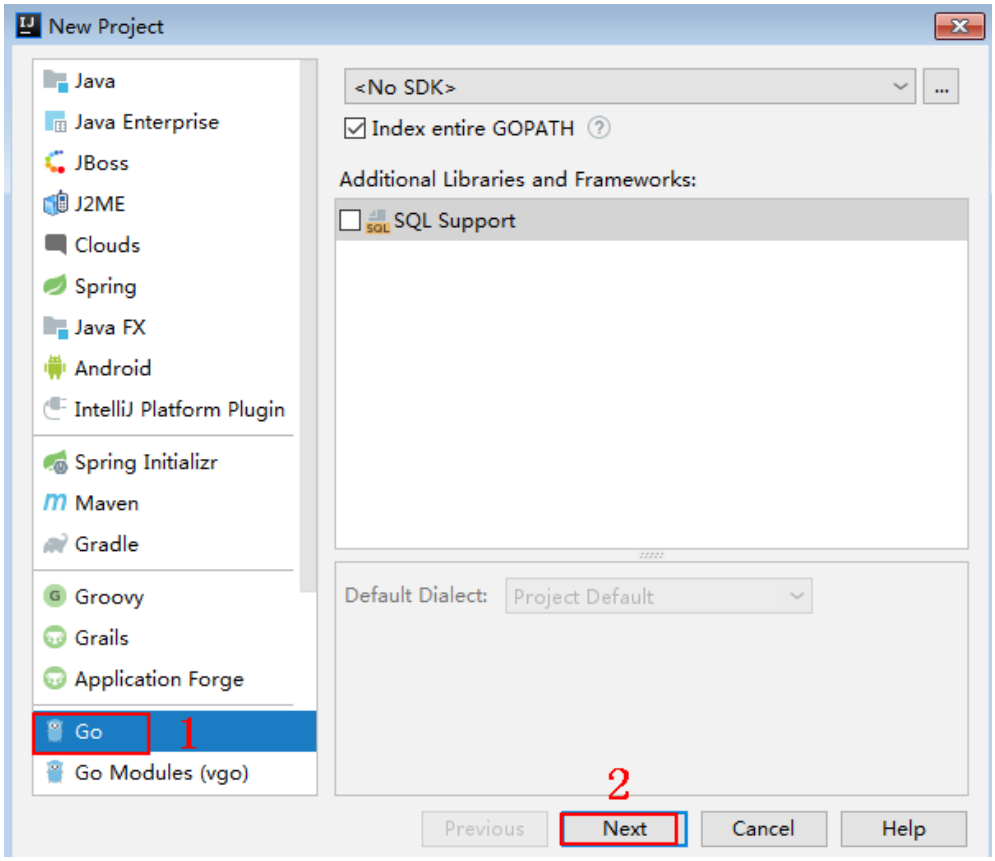
名称	说明
core\escape.go	SDK代码
core\signer.go	
demo.go	示例代码

新建工程

步骤1 打开IntelliJ IDEA，选择菜单“File > New > Project”。

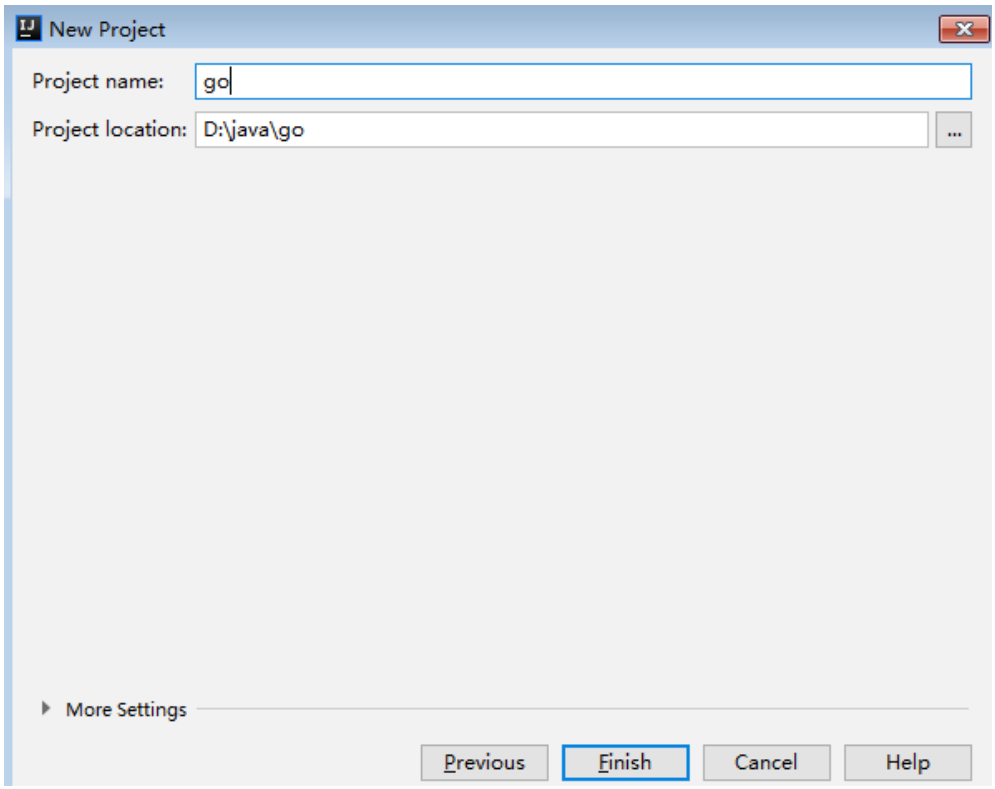
弹出“New Project”对话框，选择“Go”，单击“Next”。

图 1-18 New Project



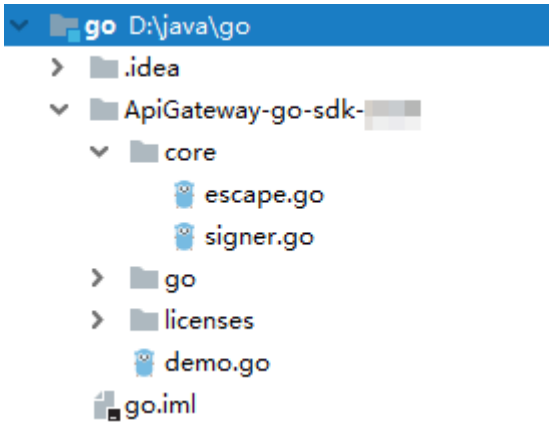
步骤2 单击“...”，在弹出的对话框中选择解压后的SDK路径，单击“Finish”。

图 1-19 选择解压后的 SDK 路径



步骤3 完成工程创建后，目录结构如下。

图 1-20 新建工程的目录结构



“demo.go”为示例代码，请根据实际情况修改参数后使用。具体代码说明请参考**调用API示例**。

----结束

调用 API 示例

步骤1 在工程中引入sdk（signer.go）。

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

步骤2 生成一个新的Signer，输入AppKey和AppSecret。

```
// 认证用的ak和sk编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
// 本示例以ak和sk保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量SDK_AK和SDK_SK。
ak = os.Getenv("SDK_AK");
sk = os.Getenv("SDK_SK");

s := core.Signer{
    Key:    ak,
    Secret: sk,
}
```

步骤3 生成一个新的Request，指定域名、方法名、请求url、query和body。

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

步骤4 给请求添加header头，内容为具体参数数据。如有需要，添加需要签名的其他头域。

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

步骤5 进行签名，执行此函数会在请求中添加用于签名的X-Sdk-Date头和Authorization头。然后为请求添加x-Authorization头，值与Authorization头相同。

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

步骤6 访问API，查看访问结果。

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

----结束

1.4.4 Python

操作场景

使用Python语言调用APP认证的API时，您需要先获取SDK，然后新建工程，最后参考调用API示例调用API。

本章节以IntelliJ IDEA 2018.3.5版本为例介绍。

准备环境

- 已获取API的域名、请求url、请求方法、AppKey和AppSecret等信息，具体参见**认证前准备**。
- 获取并安装Python安装包（可使用2.7.9+或3.X），如果未安装，请至**Python官方下载页面**下载。

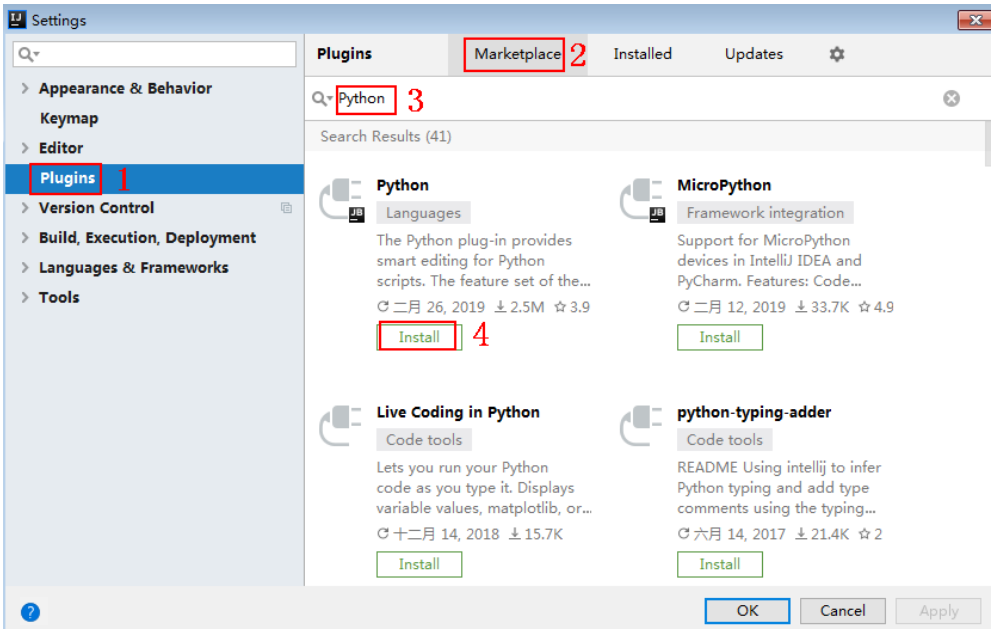
  Python安装完成后，在命令行中使用pip安装“requests”库。

  ```
  pip install requests
  ```

  说明

  如果pip安装requests遇到证书错误，请下载并使用Python执行**此文件**，升级pip，然后再执行以上命令安装。
- 获取并安装IntelliJ IDEA，如果未安装，请至**IntelliJ IDEA官方网站**下载。
- 已在IntelliJ IDEA中安装Python插件，如果未安装，请按照**图1-21**所示安装。

图 1-21 安装 Python 插件



获取 SDK

步骤1 登录DataArts Studio控制台。

步骤2 单击“数据服务”模块。

步骤3 单击左侧菜单“专享版 > SDK”。

步骤4 单击SDK使用引导区域里对应语言的SDK，下载SDK包到本地。

步骤5 进行SDK包完整性校验。Windows操作系统下，打开本地命令提示符框，输入如下命令，在本地生成已下载SDK包的SHA256值，其中，“D:\java-sdk.zip”为SDK包的本地存放路径和SDK包名，请根据实际情况修改。

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

命令执行结果示例，如下所示：

```
SHA256 的 D:\java-sdk.zip 哈希:
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
CertUtil: -hashfile 命令成功完成。
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
```

对比所下载SDK包的SHA256值和下表中对应语言SDK包的SHA256值。如果一致，则表示下载过程不存在篡改和丢包。

不同语言SDK包	SHA256值
Java	becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
Go	bcf8cf19a21226e247195f2e584c8414da39b8d05840fb02948e1375d9bbb7e6
Python	c3da3b5814f828d6217963e856563d558d938b3da28993a8a13c8a7ebff5b95d

不同语言SDK包	SHA256值
C#	a880b47e63ab35bfe216592e340a8135b866aef8f756ef7738fff3287885f33a
JavaScript	53261387f5fcf46e61d0bef5e890bea97952717f327c356412c3128389e848d6
PHP	29bf711144e77a4adaea1257cd6dedd2220e57b729a8fd000c51e68ccb42ad4b
C++	f604c6386c62cccb7c358007778037d5b15480987dc2860eef1b7bad37cb21d7
C	7086012c2d0569d5938830926b19fbea0d46682a983e04e52924978e8720c2f8
Android	89962b186707828b06b0c9f50c010b2f4cefd6a8e7ca9bdefb616bbbf6e739c8

----结束

获取“ApiGateway-python-sdk.zip”压缩包，解压后目录结构如下：

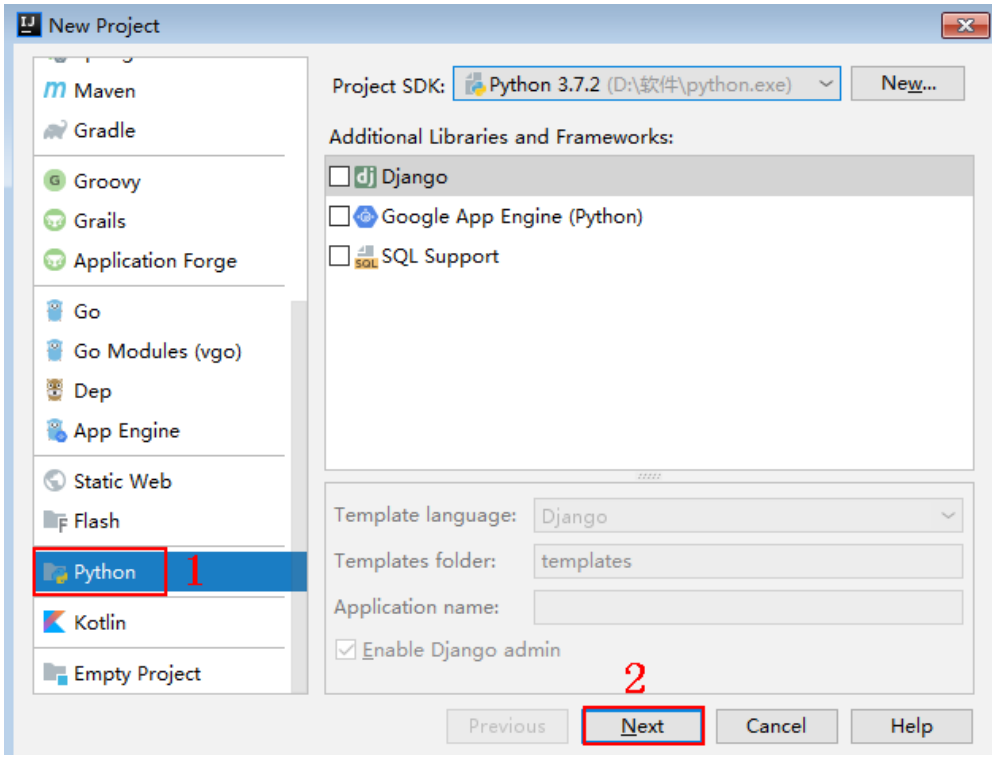
名称	说明
apig_sdk__init__.py	SDK代码
apig_sdk\signer.py	
main.py	示例代码
backend_signature.py	后端签名示例代码
licenses\license-requests	第三方库license文件

新建工程

步骤1 打开IDEA，选择菜单“File > New > Project”。

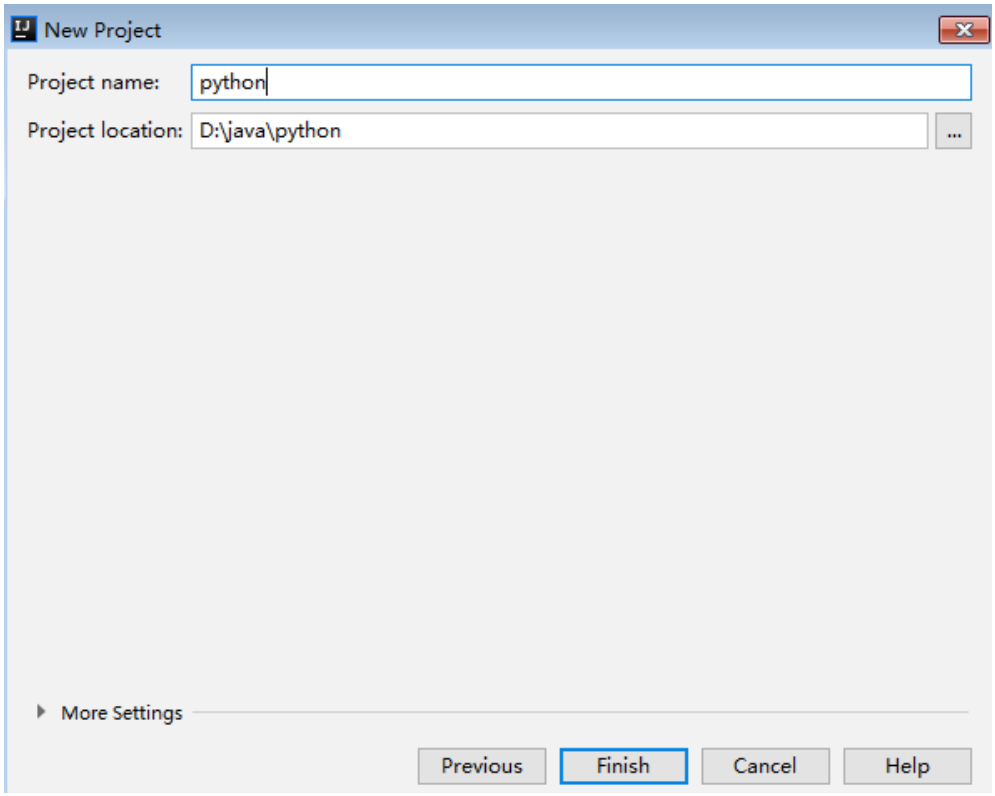
弹出“New Project”对话框，选择“Python”，单击“Next”。

图 1-22 New Project



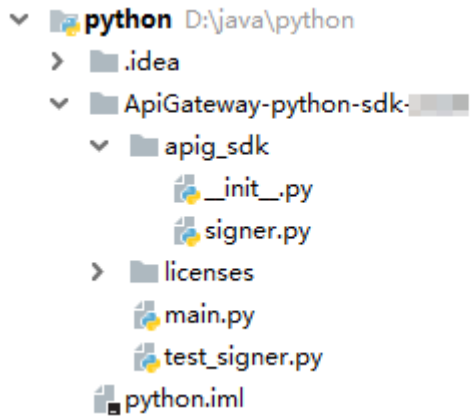
步骤2 再次单击“Next”，弹出以下对话框。单击“...”，在弹出的对话框中选择解压后的SDK路径，单击“Finish”。

图 1-23 选择解压后的 SDK 路径



步骤3 完成工程创建后，目录结构如下。

图 1-24 新建工程的目录结构



“main.py”为示例代码，请根据实际情况修改参数后使用。具体代码说明请参考**调用API示例**。

----结束

调用 API 示例

步骤1 在工程中引入apig_sdk。

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

步骤2 生成一个新的Signer，填入AppKey和AppSecret。

```
# 认证用的ak和sk编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
# 本示例以ak和sk保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量SDK_AK和SDK_SK。
ak = os.environ("SDK_AK");
sk = os.environ("SDK_SK");

sig = signer.Signer()
sig.Key = ak
sig.Secret = sk
```

步骤3 生成一个Request对象，指定方法名、请求uri、header和body。

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

步骤4 进行签名，执行此函数会在请求参数中添加用于签名的X-Sdk-Date头和Authorization头。然后为请求添加x-Authorization头，值与Authorization头相同。

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

步骤5 访问API，查看访问结果。

```
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)
```

----结束

1.4.5 C#

操作场景

使用C#语言调用APP认证的API时，您需要先获取SDK，然后打开SDK包中的工程文件，最后参考API调用示例调用API。

准备环境

- 已获取API的域名、请求url、请求方法、AppKey和AppSecret等信息，具体参见**认证前准备**。
- 获取并安装Visual Studio，如果未安装，请至**Visual Studio官方网站**下载。

获取 SDK

步骤1 登录DataArts Studio控制台。

步骤2 单击“数据服务”模块。

步骤3 单击左侧菜单“专享版 > SDK”。

步骤4 单击SDK使用引导区域里对应语言的SDK，下载SDK包到本地。

步骤5 进行SDK包完整性校验。Windows操作系统下，打开本地命令提示符框，输入如下命令，在本地生成已下载SDK包的SHA256值，其中，“D:\java-sdk.zip”为SDK包的本地存放路径和SDK包名，请根据实际情况修改。

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

命令执行结果示例，如下所示：

```
SHA256 的 D:\java-sdk.zip 哈希:
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
CertUtil: -hashfile 命令成功完成。
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
```

对比所下载SDK包的SHA256值和下表中对应语言SDK包的SHA256值。如果一致，则表示下载过程不存在篡改和丢包。

不同语言SDK包	SHA256值
Java	becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
Go	bcf8cf19a21226e247195f2e584c8414da39b8d05840fb02948e1375d9bbb7e6
Python	c3da3b5814f828d6217963e856563d558d938b3da28993a8a13c8a7ebff5b95d
C#	a880b47e63ab35bfe216592e340a8135b866aef8f756ef7738fff3287885f33a
JavaScript	53261387f5fcf46e61d0bef5e890bea97952717f327c356412c3128389e848d6
PHP	29bf711144e77a4adaea1257cd6dedd2220e57b729a8fd000c51e68ccb42ad4b

不同语言SDK包	SHA256值
C++	f604c6386c62cccb7c358007778037d5b15480987dc2860eef1b7bad37cb21d7
C	7086012c2d0569d5938830926b19fbea0d46682a983e04e52924978e8720c2f8
Android	89962b186707828b06b0c9f50c010b2f4cefd6a8e7ca9bdefb616bbbf6e739c8

----结束

获取“ApiGateway-csharp-sdk.zip”压缩包，解压后目录结构如下：

名称	说明
apigateway-signature\Signer.cs	SDK代码
apigateway-signature\HttpEncoder.cs	
sdk-request\Program.cs	签名请求示例代码
backend-signature\	后端签名示例工程
csharp.sln	工程文件
licenses\license-referencesource	第三方库license文件

打开工程

双击SDK包中的“csharp.sln”文件，打开工程。工程中包含如下3个项目：

- apigateway-signature：实现签名算法的共享库，可用于.Net Framework与.Net Core项目。
- backend-signature：后端服务签名示例。
- sdk-request：签名算法的调用示例，请根据实际情况修改参数后使用。具体代码说明请参考**调用API示例**。

调用 API 示例

步骤1 在工程中引入sdk。

```
using APIGATEWAY_SDK;
```

步骤2 生成一个新的Signer， 填入AppKey和AppSecret。

```
// 认证用的ak和sk编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
// 本示例以ak和sk保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量SDK_AK和SDK_SK。
string ak = System.Environment.GetEnvironmentVariable("SDK_AK");
```

```
string sk = System.Environment.GetEnvironmentVariable("SDK_SK");

Signer signer = new Signer();
signer.Key = ak;
signer.Secret = sk;
```

步骤3 生成一个HttpRequest对象，指定域方法名、请求url和body。

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

步骤4 给请求添加header头，内容为具体参数数据。如有需要，添加需要签名的其他头域。

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

步骤5 进行签名，执行此函数会生成一个新的HttpWebRequest，并在请求参数中添加用于签名的X-Sdk-Date头和Authorization头。然后为请求添加x-Authorization头，值与Authorization头相同。

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

步骤6 访问API，查看访问结果。

```
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());
```

----结束

1.4.6 JavaScript

操作场景

使用JavaScript语言调用APP认证的API时，您需要先获取SDK，然后新建工程，最后参考API调用示例调用API。

本章节以IntelliJ IDEA 2018.3.5版本、搭建Node.js开发环境为例介绍。

准备环境

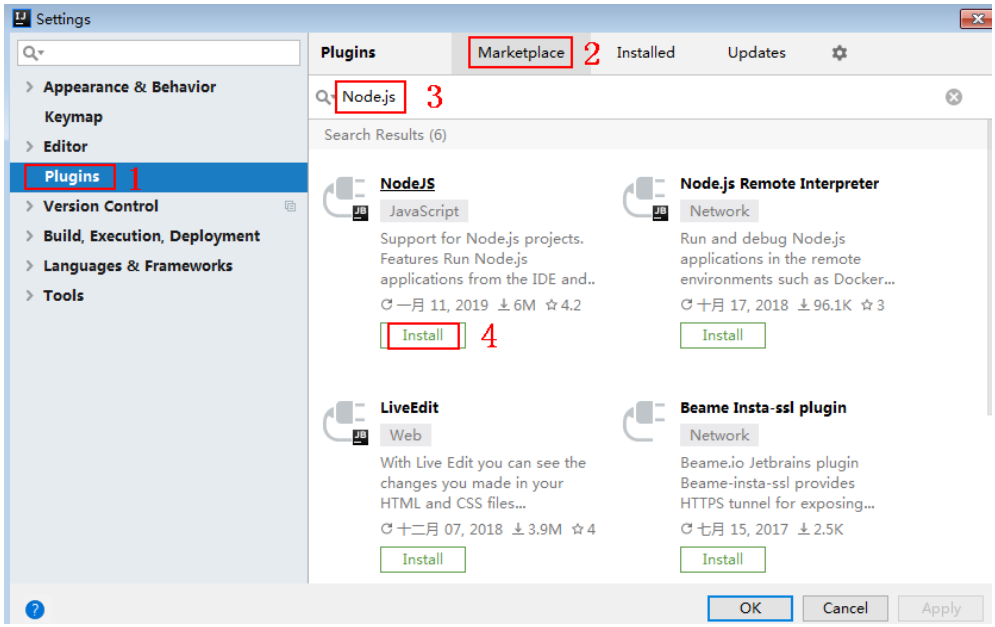
- 已获取API的域名、请求url、请求方法、AppKey和AppSecret等信息，具体参见**认证前准备**。
- 获取并安装Nodejs安装包，如果未安装，请至**Nodejs官方网站**下载。

  Nodejs安装后，在命令行中，用npm安装“moment”和“moment-timezone”模块。

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

- 获取并安装IntelliJ IDEA，如果未安装，请至**IntelliJ IDEA官方网站**下载。
- 已在IntelliJ IDEA中安装NodeJS插件，如果未安装，请按照**图1-25**所示安装。

图 1-25 安装 NodeJS 插件



获取 SDK

步骤1 登录DataArts Studio控制台。

步骤2 单击“数据服务”模块。

步骤3 单击左侧菜单“专享版 > SDK”。

步骤4 单击SDK使用引导区域里对应语言的SDK，下载SDK包到本地。

步骤5 进行SDK包完整性校验。Windows操作系统下，打开本地命令提示符框，输入如下命令，在本地生成已下载SDK包的SHA256值，其中，“D:\java-sdk.zip”为SDK包的本地存放路径和SDK包名，请根据实际情况修改。

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

命令执行结果示例，如下所示：

```
SHA256 的 D:\java-sdk.zip 哈希:
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
CertUtil: -hashfile 命令成功完成。
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
```

对比所下载SDK包的SHA256值和下表中对应语言SDK包的SHA256值。如果一致，则表示下载过程不存在篡改和丢包。

不同语言SDK包	SHA256值
Java	becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
Go	bcf8cf19a21226e247195f2e584c8414da39b8d05840fb02948e1375d9bbb7e6
Python	c3da3b5814f828d6217963e856563d558d938b3da28993a8a13c8a7ebff5b95d

不同语言SDK包	SHA256值
C#	a880b47e63ab35bfe216592e340a8135b866aef8f756ef7738fff3287885f33a
JavaScript	53261387f5fcf46e61d0bef5e890bea97952717f327c356412c3128389e848d6
PHP	29bf711144e77a4adaea1257cd6dedd2220e57b729a8fd000c51e68ccb42ad4b
C++	f604c6386c62cccb7c358007778037d5b15480987dc2860eef1b7bad37cb21d7
C	7086012c2d0569d5938830926b19fbea0d46682a983e04e52924978e8720c2f8
Android	89962b186707828b06b0c9f50c010b2f4cefd6a8e7ca9bdefb616bbbf6e739c8

----结束

获取“ApiGateway-javascript-sdk.zip”压缩包，解压后目录结构如下：

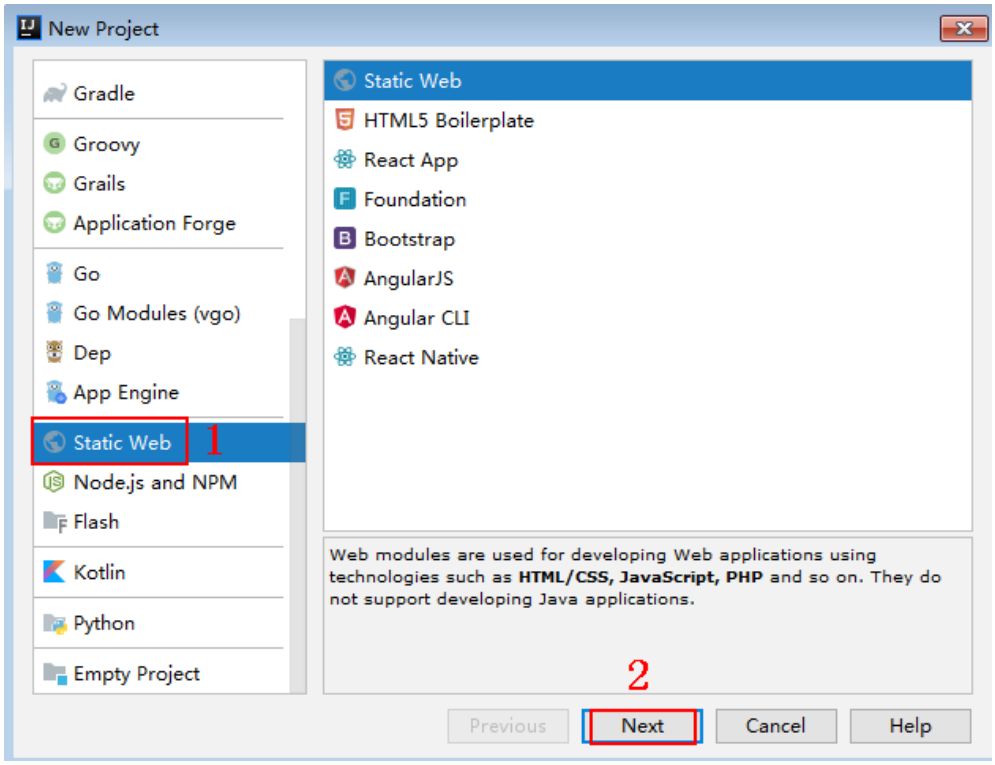
名称	说明
signer.js	SDK代码
node_demo.js	Nodejs示例代码
demo.html	浏览器示例代码
demo_require.html	浏览器示例代码（使用require加载）
test.js	测试用例
js\hmac-sha256.js	依赖库
js\moment.min.js	
js\moment-timezone-with-data.min.js	
licenses\license-crypto-js	第三方库license文件
licenses\license-moment	
licenses\license-moment-timezone	
licenses\license-node	

创建工程

步骤1 打开IntelliJ IDEA，选择菜单“File > New > Project”。

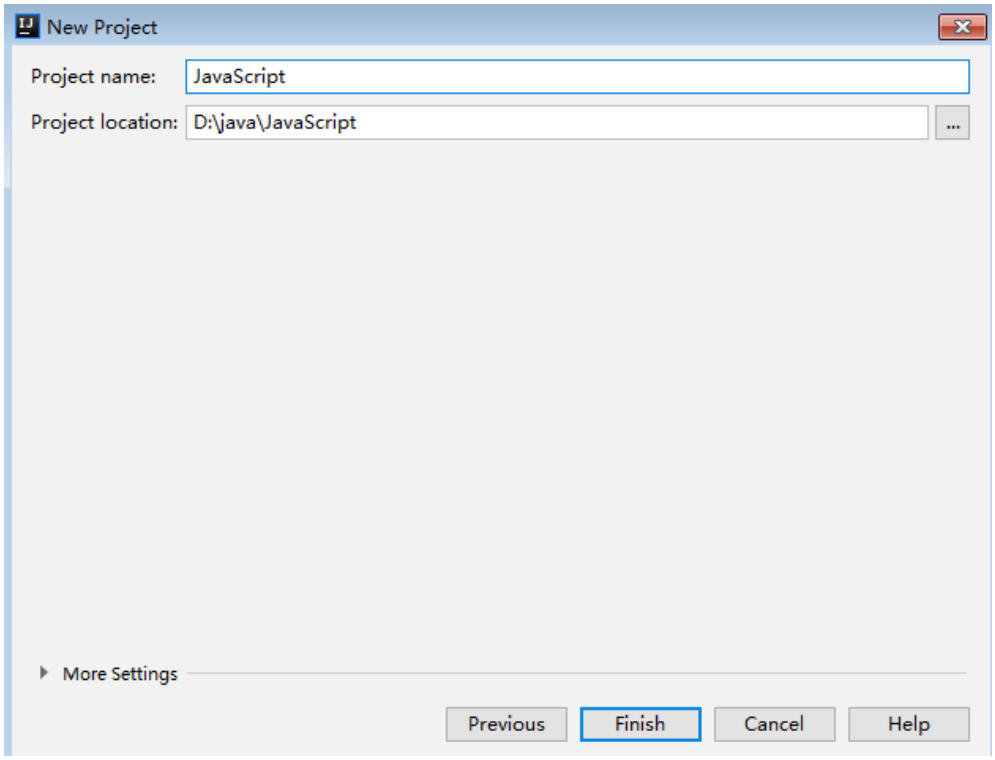
弹出“New Project”对话框。选择“Static Web”，单击“Next”。

图 1-26 New Project



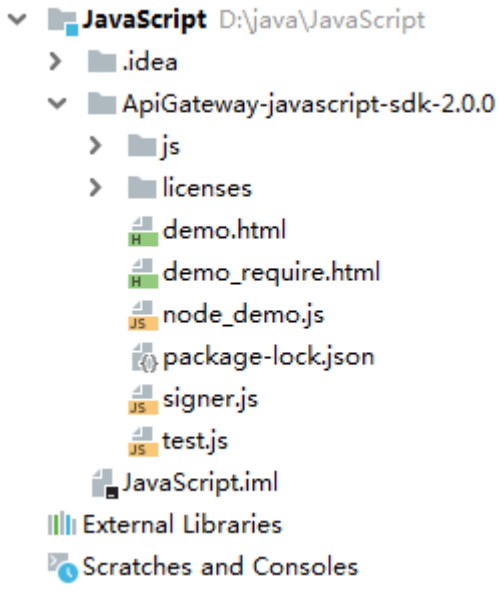
步骤2 单击“...”，在弹出的对话框中选择解压后的SDK路径，单击“Finish”。

图 1-27 选择解压后的 SDK 路径



步骤3 完成工程创建后，目录结构如下。

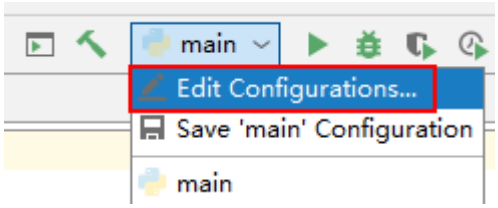
图 1-28 新建工程的目录结构



- node_demo.js：Nodejs示例代码，请根据实际情况修改参数后使用。具体代码说明请参考**调用API（Node.js）示例**。

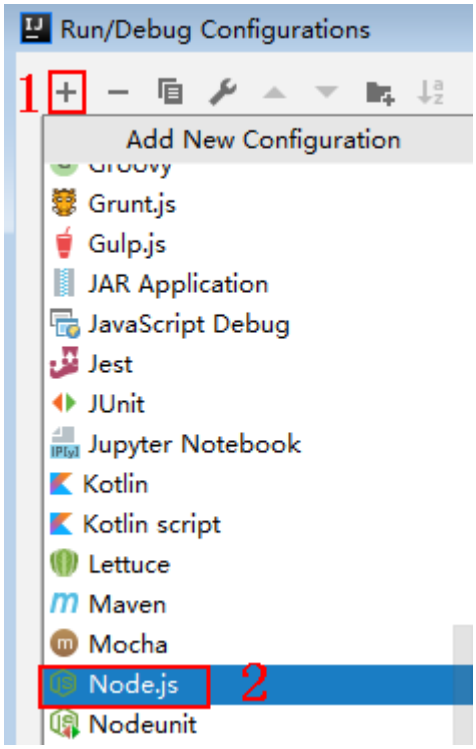
步骤4 单击“Edit Configurations”，弹出“Run/Debug Configurations”对话框。

图 1-29 Edit Configurations



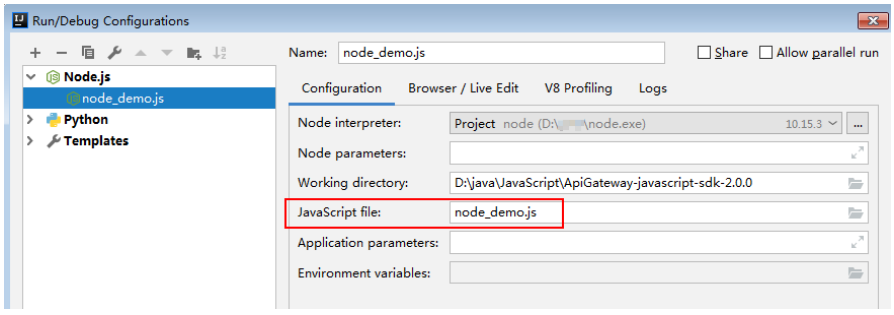
步骤5 单击“+”，选择“Node.js”。

图 1-30 选择 Node.js



步骤6 “JavaScript file”选择“node_demo.js”，单击“OK”，完成配置。

图 1-31 选择 node_demo.js



----结束

调用 API（Node.js）示例

步骤1 在工程中引入signer.js。

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

步骤2 生成一个新的Signer，填入AppKey和AppSecret。

```
// 认证用的ak和sk编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
// 本示例以ak和sk保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量SDK_AK和SDK_SK。
var ak = process.env.SDK_AK;
var sk = process.env.SDK_SK;

var sig = new signer.Signer();
sig.Key = ak;
sig.Secret = sk;
```

步骤3 生成一个Request对象，指定方法名、请求uri和body。

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

步骤4 给请求添加header头，内容为具体参数数据。如有需要，添加需要签名的其他头域。

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

步骤5 进行签名，执行此函数会生成请求参数，用于创建http(s)请求，请求参数中添加了用于签名的X-Sdk-Date头和Authorization头。然后为请求参数添加x-Authorization头，值与Authorization头相同。

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

步骤6 访问API，查看访问结果。如果使用https访问，则将“http.request”改为“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()
```

----结束

1.4.7 PHP

操作场景

使用PHP语言调用APP认证的API时，您需要先获取SDK，然后新建工程，最后参考**调用API示例**调用API。

本章节以IntelliJ IDEA 2018.3.5版本为例介绍。

准备环境

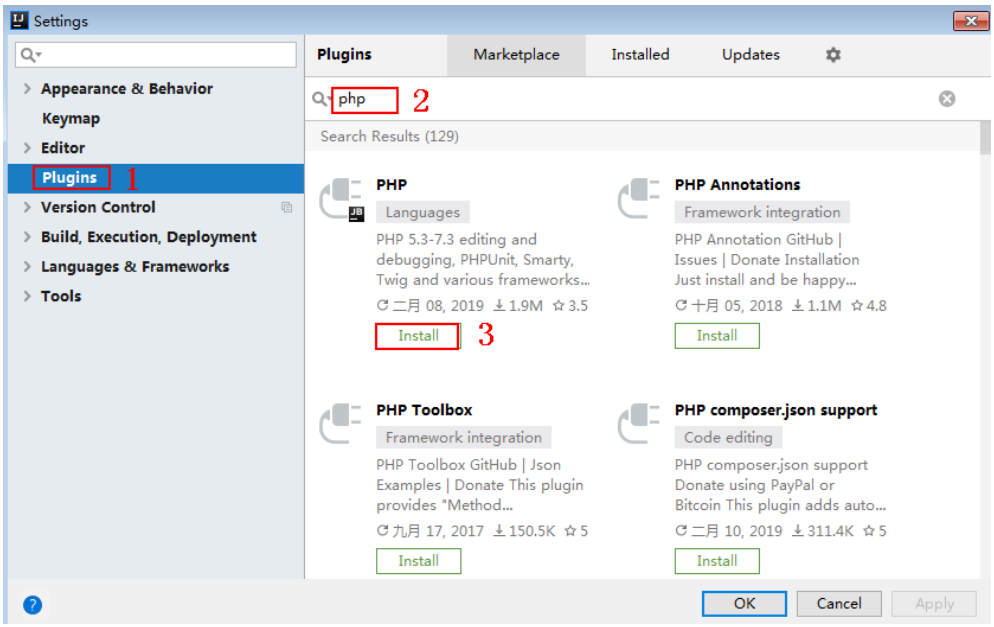
- 已获取API的域名、请求url、请求方法、AppKey和AppSecret等信息，具体参见**认证前准备**。

- 获取并安装IntelliJ IDEA，如果未安装，请至**IntelliJ IDEA官方网站**下载。
- 获取并安装PHP安装包，如果未安装，请至**PHP官方下载页面**下载。
- 将PHP安装目录中的“php.ini-production”文件复制到“C:\windows”，改名为“php.ini”，并在文件中增加如下内容。

```
extension_dir = "php安装目录/ext"
extension=openssl
extension=curl
```

- 已在IntelliJ IDEA中安装PHP插件，如果未安装，请按照**图1-32**所示安装。

图 1-32 安装 PHP 插件



获取 SDK

步骤1 登录DataArts Studio控制台。

步骤2 单击“数据服务”模块。

步骤3 单击左侧菜单“专享版 > SDK”。

步骤4 单击SDK使用引导区域里对应语言的SDK，下载SDK包到本地。

步骤5 进行SDK包完整性校验。Windows操作系统下，打开本地命令提示符框，输入如下命令，在本地生成已下载SDK包的SHA256值，其中，“D:\java-sdk.zip”为SDK包的本地存放路径和SDK包名，请根据实际情况修改。

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

命令执行结果示例，如下所示：

```
SHA256 的 D:\java-sdk.zip 哈希:
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
CertUtil: -hashfile 命令成功完成。
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
```

对比所下载SDK包的SHA256值和下表中对应语言SDK包的SHA256值。如果一致，则表示下载过程不存在篡改和丢包。

不同语言SDK包	SHA256值
Java	becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
Go	bcf8cf19a21226e247195f2e584c8414da39b8d05840fb02948e1375d9bbb7e6
Python	c3da3b5814f828d6217963e856563d558d938b3da28993a8a13c8a7ebff5b95d
C#	a880b47e63ab35bfe216592e340a8135b866aef8f756ef7738fff3287885f33a
JavaScript	53261387f5fcf46e61d0bef5e890bea97952717f327c356412c3128389e848d6
PHP	29bf711144e77a4adaea1257cd6dedd2220e57b729a8fd000c51e68ccb42ad4b
C++	f604c6386c62cccb7c358007778037d5b15480987dc2860eef1b7bad37cb21d7
C	7086012c2d0569d5938830926b19fbea0d46682a983e04e52924978e8720c2f8
Android	89962b186707828b06b0c9f50c010b2f4cefd6a8e7ca9bdefb616bbbf6e739c8

----结束

获取“ApiGateway-php-sdk.zip”压缩包，解压后目录结构如下：

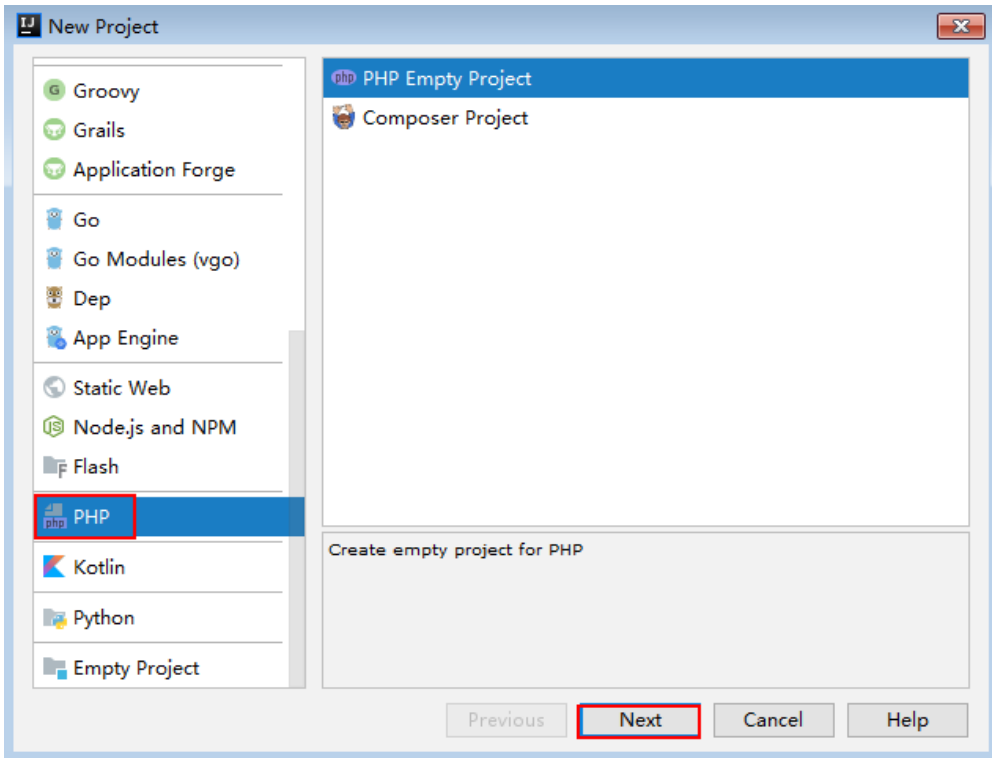
名称	说明
signer.php	SDK代码
index.php	示例代码

新建工程

步骤1 打开IDEA，选择菜单“File > New > Project”。

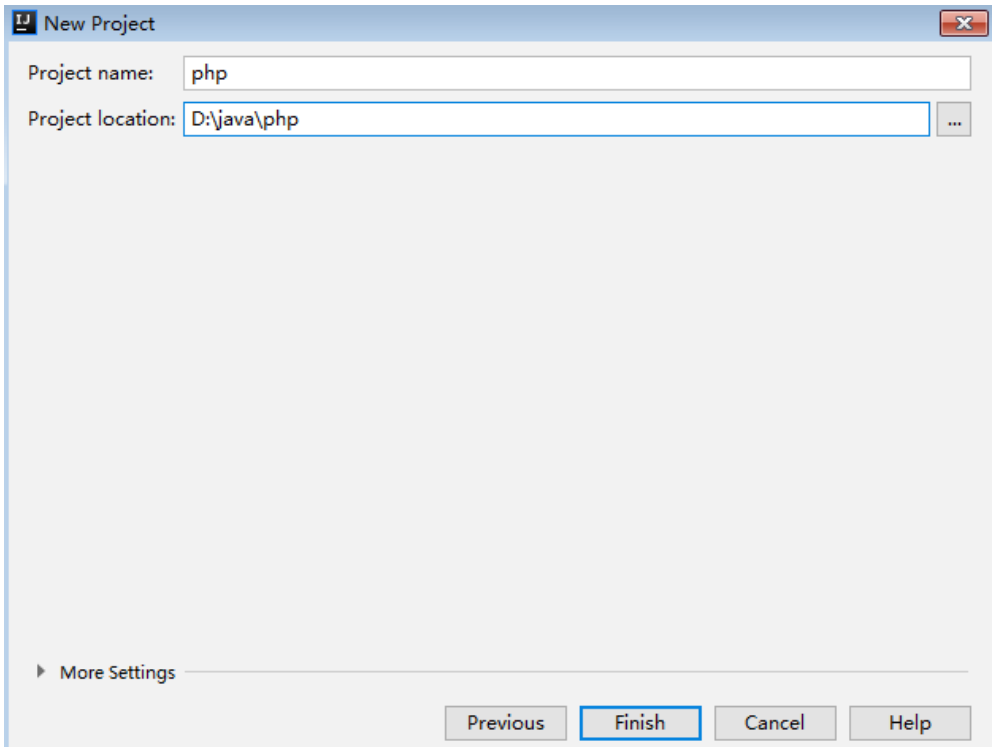
弹出“New Project”对话框，选择“PHP”，单击“Next”。

图 1-33 New Project



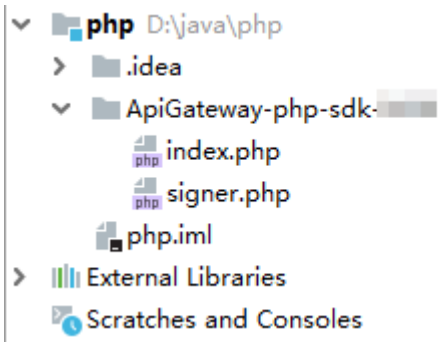
步骤2 单击“...”，在弹出的对话框中选择解压后的SDK路径，单击“Finish”。

图 1-34 选择解压后的 SDK 路径



步骤3 完成工程创建后，目录结构如下。

图 1-35 新建工程的目录结构



“signer.php”为示例代码，请根据实际情况修改参数后使用。具体代码说明请参考**调用API示例**。

----结束

调用 API 示例

步骤1 在代码中引入sdk。

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

步骤2 生成一个新的Signer， 填入AppKey和AppSecret。

```
// 认证用的ak和sk编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
// 本示例以ak和sk保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量SDK_AK和SDK_SK。
$ak = getenv('SDK_AK');
$sk = getenv("SDK_SK");

$signer = new Signer();
$signer->Key = $ak;
$signer->Secret = $sk;
```

步骤3 生成一个新的Request，指定方法名、请求url和body。

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

步骤4 给请求添加header头，内容为具体参数数据。如果有需要，添加需要签名的其他头域。其中host为必填项，需要填入url的请求地址，示例代码如“host”作为样例。

```
$req->headers = array(
    'host' => 'serviceEndpoint'
);
```

步骤5 进行签名，执行此函数会生成一个$curl上下文变量。然后为请求添加x-Authorization头，值与Authorization头相同。

```
$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);
```

步骤6 访问API，查看访问结果。

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

```
echo $response;
curl_close($curl);
```

----结束

1.4.8 C++

操作场景

使用C++语言调用APP认证的API时，您需要先获取SDK，参考API调用示例调用API。

准备环境

1. 已获取API的域名、请求url、请求方法、AppKey和AppSecret等信息，具体参见**认证前准备**。
2. 安装openssl库。
   ```
   apt-get install libssl-dev
   ```
3. 安装curl库。
   ```
   apt-get install libcurl4-openssl-dev
   ```

获取 SDK

步骤1 登录DataArts Studio控制台。

步骤2 单击“数据服务”模块。

步骤3 单击左侧菜单“专享版 > SDK”。

步骤4 单击SDK使用引导区域里对应语言的SDK，下载SDK包到本地。

步骤5 进行SDK包完整性校验。Windows操作系统下，打开本地命令提示符框，输入如下命令，在本地生成已下载SDK包的SHA256值，其中，“D:\java-sdk.zip”为SDK包的本地存放路径和SDK包名，请根据实际情况修改。

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

命令执行结果示例，如下所示：

```
SHA256 的 D:\java-sdk.zip 哈希:
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
CertUtil: -hashfile 命令成功完成。
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
```

对比所下载SDK包的SHA256值和下表中对应语言SDK包的SHA256值。如果一致，则表示下载过程不存在篡改和丢包。

不同语言SDK包	SHA256值
Java	becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
Go	bcf8cf19a21226e247195f2e584c8414da39b8d05840fb02948e1375d9bbb7e6
Python	c3da3b5814f828d6217963e856563d558d938b3da28993a8a13c8a7ebff5b95d

不同语言SDK包	SHA256值
C#	a880b47e63ab35bfe216592e340a8135b866aef8f756ef7738fff3287885f33a
JavaScript	53261387f5fcf46e61d0bef5e890bea97952717f327c356412c3128389e848d6
PHP	29bf711144e77a4adaea1257cd6dedd2220e57b729a8fd000c51e68ccb42ad4b
C++	f604c6386c62cccb7c358007778037d5b15480987dc2860eef1b7bad37cb21d7
C	7086012c2d0569d5938830926b19fbea0d46682a983e04e52924978e8720c2f8
Android	89962b186707828b06b0c9f50c010b2f4cefd6a8e7ca9bdefb616bbbf6e739c8

----结束

获取“ApiGateway-cpp-sdk.zip”压缩包，解压后目录结构如下：

名称	说明
hasher.cpp	SDK代码
hasher.h	
header.h	
RequestParams.cpp	
RequestParams.h	
signer.cpp	
signer.h	
Makefile	Makefile文件
main.cpp	示例代码

调用 API 示例

步骤1 在main.cpp中加入以下引用。

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

步骤2 生成一个新的Signer， 填入AppKey和AppSecret。

```
// 认证用的ak和sk编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
```

```
// 本示例以ak和sk保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量SDK_AK和SDK_SK。
char* ak = getenv("SDK_AK");
char* sk = getenv("SDK_SK");
Signer signer(ak, sk);
```

步骤3 生成一个新的RequestParams，指定方法名、域名、请求uri、查询字符串和body。

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

步骤4 给请求添加header头，内容为具体参数数据。如果有需要，添加需要签名的其他头域。

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

步骤5 进行签名，执行此函数会将生成的签名头加入request变量中。然后为请求添加x-Authorization头，值与Authorization头相同。

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

步骤6 使用curl库访问API，查看访问结果。

```
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;
}
```

步骤7 运行make命令编译，得到可执行文件main，执行main文件，查看结果。

----结束

1.4.9 C

操作场景

使用C语言调用APP认证的API时，您需要先获取SDK，参考API调用示例调用API。

准备环境

1. 已获取API的域名、请求url、请求方法、AppKey和AppSecret等信息，具体参见**认证前准备**。
2. 安装openssl库。
   ```
   apt-get install libssl-dev
   ```
3. 安装curl库。
   ```
   apt-get install libcurl4-openssl-dev
   ```

获取 SDK

步骤1 登录DataArts Studio控制台。

步骤2 单击“数据服务”模块。

步骤3 单击左侧菜单“专享版 > SDK”。

步骤4 单击SDK使用引导区域里对应语言的SDK，下载SDK包到本地。

步骤5 进行SDK包完整性校验。Windows操作系统下，打开本地命令提示符框，输入如下命令，在本地生成已下载SDK包的SHA256值，其中，“D:\java-sdk.zip”为SDK包的本地存放路径和SDK包名，请根据实际情况修改。

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

命令执行结果示例，如下所示：

```
SHA256 的 D:\java-sdk.zip 哈希:
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
CertUtil: -hashfile 命令成功完成。
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
```

对比所下载SDK包的SHA256值和下表中对应语言SDK包的SHA256值。如果一致，则表示下载过程不存在篡改和丢包。

不同语言SDK包	SHA256值
Java	becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
Go	bcf8cf19a21226e247195f2e584c8414da39b8d05840fb02948e1375d9bbb7e6
Python	c3da3b5814f828d6217963e856563d558d938b3da28993a8a13c8a7ebff5b95d
C#	a880b47e63ab35bfe216592e340a8135b866aef8f756ef7738fff3287885f33a
JavaScript	53261387f5fcf46e61d0bef5e890bea97952717f327c356412c3128389e848d6
PHP	29bf711144e77a4adaea1257cd6dedd2220e57b729a8fd000c51e68ccb42ad4b
C++	f604c6386c62cccb7c358007778037d5b15480987dc2860eef1b7bad37cb21d7
C	7086012c2d0569d5938830926b19fbea0d46682a983e04e52924978e8720c2f8
Android	89962b186707828b06b0c9f50c010b2f4cefd6a8e7ca9bdefb616bbbf6e739c8

----结束

获取“ApiGateway-c-sdk.zip”压缩包，解压后目录结构如下：

名称	说明
signer_common.c	SDK代码
signer_common.h	
signer.c	
signer.h	
Makefile	Makefile文件
main.c	示例代码

调用 API 示例

步骤1 在main.c中加入以下引用。

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

步骤2 生成一个sig_params_t类型的变量， 填入AppKey和AppSecret。

```
sig_params_t params;
sig_params_init(&params);

// 认证用的ak和sk编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
// 本示例以ak和sk保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量SDK_AK和SDK_SK。
char* ak = getenv("SDK_AK");
char* sk = getenv("SDK_SK");

sig_str_t app_key = sig_str(ak);
sig_str_t app_secret = sig_str(sk);
params.key = app_key;
params.secret = app_secret;
```

步骤3 指定方法名、域名、请求uri、查询字符串和body。

```
sig_str_t host = sig_str("serviceEndpoint");
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;
```

步骤4 给请求添加header头，内容为具体参数数据。如果有需要，添加需要签名的其他头域。

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

步骤5 进行签名，执行此函数会将生成的签名头加入request变量中。然后为请求添加x-Authorization头，值与Authorization头相同。

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

步骤6 使用curl库访问API，查看访问结果。

```
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;
}
```

步骤7 运行make命令编译，得到可执行文件main，执行main文件，查看结果。

----结束

1.4.10 Android

操作场景

使用Android语言调用APP认证的API时，您需要先获取SDK，然后新建工程，最后参考API调用示例调用API。

准备环境

- 已获取API的域名、请求url、请求方法、AppKey和AppSecret等信息，具体参见**认证前准备**。
- 获取并安装Android Studio，如果未安装，请至**Android Studio官方网站**下载。

获取 SDK

步骤1 登录DataArts Studio控制台。

步骤2 单击“数据服务”模块。

步骤3 单击左侧菜单“专享版 > SDK”。

步骤4 单击SDK使用引导区域里对应语言的SDK，下载SDK包到本地。

步骤5 进行SDK包完整性校验。Windows操作系统下，打开本地命令提示符框，输入如下命令，在本地生成已下载SDK包的SHA256值，其中，“D:\java-sdk.zip”为SDK包的本地存放路径和SDK包名，请根据实际情况修改。

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

命令执行结果示例，如下所示：

```
SHA256 的 D:\java-sdk.zip 哈希:
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
CertUtil: -hashfile 命令成功完成。
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
```

对比所下载SDK包的SHA256值和下表中对应语言SDK包的SHA256值。如果一致，则表示下载过程不存在篡改和丢包。

不同语言SDK包	SHA256值
Java	becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
Go	bcf8cf19a21226e247195f2e584c8414da39b8d05840fb02948e1375d9bbb7e6
Python	c3da3b5814f828d6217963e856563d558d938b3da28993a8a13c8a7ebff5b95d
C#	a880b47e63ab35bfe216592e340a8135b866aef8f756ef7738fff3287885f33a
JavaScript	53261387f5fcf46e61d0bef5e890bea97952717f327c356412c3128389e848d6
PHP	29bf711144e77a4adaea1257cd6dedd2220e57b729a8fd000c51e68ccb42ad4b

不同语言SDK包	SHA256值
C++	f604c6386c62cccb7c358007778037d5b15480987dc2860eef1b7bad37cb21d7
C	7086012c2d0569d5938830926b19fbea0d46682a983e04e52924978e8720c2f8
Android	89962b186707828b06b0c9f50c010b2f4cefd6a8e7ca9bdefb616bbbf6e739c8

----结束

获取“ApiGateway-android-sdk.zip”压缩包，解压后目录结构如下：

名称	说明
app\	安卓工程代码
gradle\	gradle相关文件
build.gradle	gradle配置文件
gradle.properties	
settings.gradle	
gradlew	gradle wrapper执行脚本
gradlew.bat	

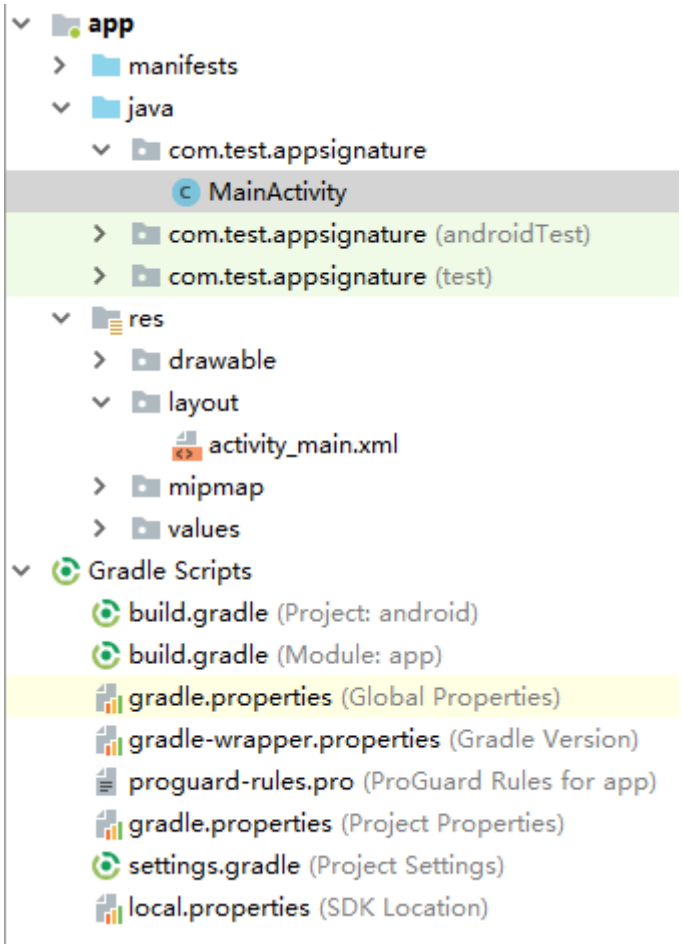
打开工程

步骤1 打开Android Studio，选择“File > Open”。

在弹出的对话框中选择解压后的SDK路径。

步骤2 打开工程后，目录结构如下。

图 1-36 工程目录结构



----结束

调用 API 示例

步骤1 在Android工程中的“app/libs”目录下，加入SDK所需jar包。其中jar包必须包括：

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

步骤2 在“build.gradle”文件中加入okhttp库的依赖。

在“build.gradle”文件中的“dependencies”下加入“implementation 'com.squareup.okhttp3:okhttp:3.11.0'”。

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

步骤3 创建request，输入AppKey和AppSecret，并指定域名、方法名、请求uri和body。

```
Request request = new Request();
try {
        // 认证用的ak和sk编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
        // 本示例以ak和sk保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量
```

```
SDK_AK和SDK_SK。
        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");

        request.setKey(ak);
        request.setSecret(sk);

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

步骤4 对请求进行签名，并为请求添加x-Authorization头，值与Authorization头相同。然后生成okhttp3.Request对象来访问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();
```

----结束

1.4.11 curl

操作场景

使用curl命令调用APP认证的API时，您需要先下载JavaScript SDK生成curl命令，然后将curl命令复制到命令行调用API。

前提条件

已获取API的域名、请求url、请求方法、AppKey和AppSecret等信息，具体参见**认证前准备**。

获取 SDK

步骤1 登录DataArts Studio控制台。

步骤2 单击“数据服务”模块。

步骤3 单击左侧菜单“专享版 > SDK”。

步骤4 单击SDK使用引导区域里对应语言的SDK，下载SDK包到本地。

步骤5 进行SDK包完整性校验。Windows操作系统下，打开本地命令提示符框，输入如下命令，在本地生成已下载SDK包的SHA256值，其中，“D:\java-sdk.zip”为SDK包的本地存放路径和SDK包名，请根据实际情况修改。

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

命令执行结果示例，如下所示：

```
SHA256 的 D:\java-sdk.zip 哈希:
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
CertUtil: -hashfile 命令成功完成。
becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
```

对比所下载SDK包的SHA256值和下表中对应语言SDK包的SHA256值。如果一致，则表示下载过程不存在篡改和丢包。

不同语言SDK包	SHA256值
Java	becff4310645f3734344897ffdcabb1853d4b7d93b59a6ea187c5ae40543b36b
Go	bcf8cf19a21226e247195f2e584c8414da39b8d05840fb02948e1375d9bbb7e6
Python	c3da3b5814f828d6217963e856563d558d938b3da28993a8a13c8a7ebff5b95d
C#	a880b47e63ab35bfe216592e340a8135b866aef8f756ef7738fff3287885f33a
JavaScript	53261387f5fcf46e61d0bef5e890bea97952717f327c356412c3128389e848d6
PHP	29bf711144e77a4adaea1257cd6dedd2220e57b729a8fd000c51e68ccb42ad4b
C++	f604c6386c62cccb7c358007778037d5b15480987dc2860eef1b7bad37cb21d7
C	7086012c2d0569d5938830926b19fbea0d46682a983e04e52924978e8720c2f8
Android	89962b186707828b06b0c9f50c010b2f4cefd6a8e7ca9bdefb616bbbf6e739c8

----结束

获取“ApiGateway-javascript-sdk.zip”压缩包，解压后目录结构如下：

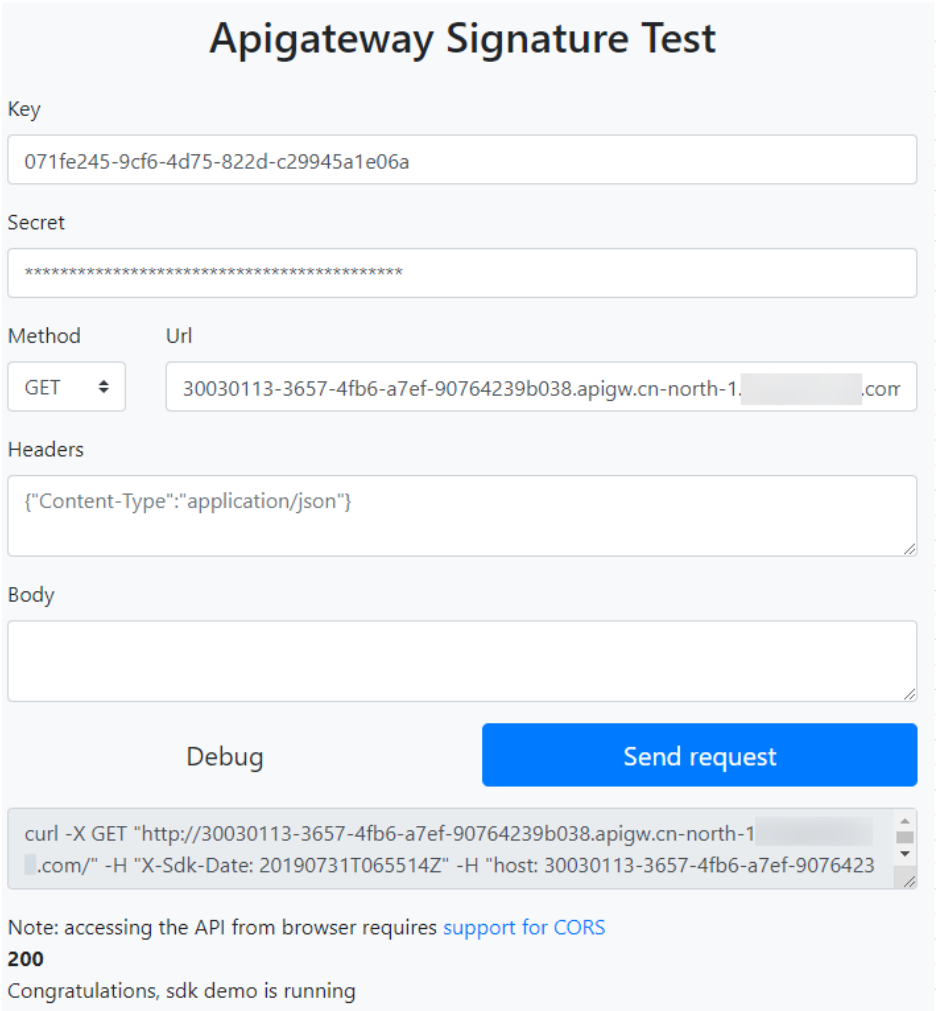
名称	说明
signer.js	SDK代码
node_demo.js	Nodejs示例代码
demo.html	浏览器示例代码
demo_require.html	浏览器示例代码（使用require加载）
test.js	测试用例
js\hmac-sha256.js	依赖库
js\moment.min.js	
js\moment-timezone-with-data.min.js	
licenses\license-crypto-js	第三方库license文件

名称	说明
licenses\license-moment	
licenses\license-moment-timezone	
licenses\license-node	

调用 API 示例

步骤1 使用JavaScript SDK生成curl命令。

获取“ApiGateway-javascript-sdk.zip”压缩包并解压。在浏览器中打开demo.html，页面如下图所示。



步骤2 填入Key、Secret、方法名、请求协议、域名和url（认证用的ak和sk编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全）。本示例从前端输入，仅用于演示，例如：

```
Key=4f5f626b-073f-402f-a1e0-e52171c6100c
Secret=******
```

```
Method=POST
Url=https://serviceEndpoint
```

步骤3 填入json格式的Query和Headers，填入Body。其中所访问API的ID为必填项，需要填入具体的ID信息，以参数"x-api-id"填入Headers中。

步骤4 单击“Send request”，生成curl命令。

```
$ curl -X POST "https://serviceEndpoint/" -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 ""
```

步骤5 为命令添加x-Authorization头，值与Authorization头相同。将curl命令复制到命令行，访问API。

```
$ curl -X POST "https://serviceEndpoint/" -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
```

----结束

1.4.12 其他编程语言

APP 认证工作原理

1. **构造规范请求**。

   将待发送的请求内容按照与API网关后台约定的规则组装，确保客户端签名、API网关后台认证时使用的请求内容一致。

2. 使用规范请求和其他信息**创建待签字符串**。
3. 使用AK/SK和待签字符串**计算签名**。
4. 将生成的**签名信息作为请求消息头添加**到HTTP请求中，或者作为查询字符串参数添加到HTTP请求中。
5. API网关收到请求后，执行**1**~**3**，计算签名。
6. 将**3**中的生成的签名与**5**中生成的签名进行比较，如果签名匹配，则处理请求，否则将拒绝请求。

说明

APP签名仅支持Body体12M及以下的请求签名。

步骤 1：构造规范请求

使用APP方式进行签名与认证，首先需要规范请求内容，然后再进行签名。客户端与API网关使用相同的请求规范，可以确保同一个HTTP请求的前后端得到相同的签名结果，从而完成身份校验。

HTTP请求规范伪代码如下：

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

可以通过以下示例来说明规范请求的构造步骤。

假设原始请求如下：

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

1. 构造**HTTP请求方法**（**HTTPRequestMethod**），以换行符结束。

   HTTP请求方法，如GET、PUT、POST等。请求方法示例：

   ```
   GET
   ```

2. 添加规范**URI参数**（**CanonicalURI**），以换行符结束。

   释义：

   规范URI，即请求资源路径，是URI的绝对路径部分的URI编码。

   格式：

   根据RFC 3986标准化URI路径，移除冗余和相对路径部分，路径中每个部分必须为URI编码。如果URI路径不以“/”结尾，则在尾部添加“/”。

   举例：

   示例中的URI：/app1，此时规范的URI编码为：

   ```
   GET
   /app1/
   ```

   说明

   计算签名时，URI必须以“/”结尾。发送请求时，可以不以“/”结尾。

3. 添加**规范查询字符串（CanonicalQueryString）**，以换行符结束。

   释义：

   查询字符串，即查询参数。如果没有查询参数，则为空字符串，即规范后的请求为**空行**。

   格式：

   规范查询字符串需要满足以下要求：

   - 根据以下规则对每个参数名和值进行URI编码：
     - 请勿对RFC 3986定义的任何非预留字符进行URI编码，这些字符包括：A-Z、a-z、0-9、-、_、.和~。
     - 使用%XY对所有非预留字符进行百分比编码，其中X和Y为十六进制字符（0-9和A-F）。例如，空格字符必须编码为%20，扩展UTF-8字符必须采用“%XY%ZA%BC”格式。
   - 对于每个参数，追加“*URI编码的参数名称*=*URI编码的参数值*”。如果没有参数值，则以空字符串代替，但不能省略“=”。

     例如以下含有两个参数，其中第二个参数parm2的值为空。

     ```
     parm1=value1&parm2=
     ```
   - 按照字符代码以升序顺序对参数名进行排序。例如，以大写字母F开头的参数名排在以小写字母b开头的参数名之前。
   - 以排序后的第一个参数名开始，构造规范查询字符串。

   举例：

   示例中包含两个可选参数：a、b

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

4. 添加规范**消息头**（**CanonicalHeaders**），以换行符结束。

   释义：

   规范消息头，即请求消息头列表。包括签名请求中的所有HTTP消息头列表。消息头必须包含X-Sdk-Date，用于校验签名时间，格式为ISO8601标准的UTC时间格式：YYYYMMDDTHHMMSSZ。如果API发布到非RELEASE环境时，需要增加自定义的环境名称。

   格式：

   CanonicalHeaders由多个请求消息头共同组成，**CanonicalHeadersEntry0 + CanonicalHeadersEntry1 + ...**，其中每个请求消息头（CanonicalHeadersEntry）**的格式为Lowercase(HeaderName) + ':' + Trimall(HeaderValue) + '\n'**

   说明

   - Lowercase表示将所有字符转换为小写字母的函数。
   - Trimall表示删除值前后的多余空格的函数。
   - 最后一个请求消息头也会携带一个换行符。叠加规范中**CanonicalHeaders**自身携带的换行符，因此会出现一个空行。

   举例：

   ```
   GET
   /app1/
   a=1&b=2
   host:serviceEndpoint
   x-sdk-date:20180330T123600Z
   ```

   须知

   规范消息头需要满足以下要求：

   - 将消息头名称转换为小写形式，并删除前导空格和尾随空格。
   - 按照字符代码对消息头名称进行升序排序。

   例如原始消息头为：

   ```
   Host: serviceEndpoint\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
   ```

   规范消息头为：

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

5. 添加用于**签名的消息头声明**（**SignedHeaders**），以换行符结束。

   释义：

   用于签名的请求消息头列表。通过添加此消息头，向API网关告知请求中哪些消息头是签名过程的一部分，以及在验证请求时API网关可以忽略哪些消息头。X-Sdk-date必须作为已签名的消息头。

   格式：

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

已签名的消息头需要满足以下要求：将已签名的消息头名称转换为小写形式，按照字符代码对消息头进行排序，并使用“;”来分隔多个消息头。

Lowercase表示将所有字符转换为小写字母。

举例：

以下表示有两个消息头参与签名：host、x-sdk-date

```
GET
/app1/
a=1&b=2
host:serviceEndpoint
x-sdk-date:20180330T123600Z

host;x-sdk-date
```

6. 使用SHA 256哈希函数以基于HTTP或HTTPS请求正文中的body体（**RequestPayload**），创建哈希值。

   释义：

   请求消息体。消息体需要做两层转换：HexEncode(Hash(*RequestPayload*))，其中Hash表示生成消息摘要的函数，当前支持SHA-256算法。HexEncode表示以小写字母形式返回摘要的Base-16编码的函数。例如，HexEncode("m") 返回值为“6d”而不是“6D”。输入的每一个字节都表示为两个十六进制字符。

   说明

   计算RequestPayload的哈希值时，对于“RequestPayload==null”的场景，直接使用空字符串""来计算。

   举例：

   本示例为GET方法，body体为空。经过哈希处理的body（空字符串）如下：

   ```
   GET
   /app1/
   a=1&b=2
   host:serviceEndpoint
   x-sdk-date:20180330T123600Z

   host;x-sdk-date
   e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
   ```

7. 对构造好的规范请求进行哈希处理，算法与对RequestPayload哈希处理的算法相同。经过哈希处理的规范请求必须以小写十六进制字符串形式表示。

   算法伪代码：**Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))**

   经过哈希处理的规范请求示例：

   ```
   4bd8e1afe76738a332ecff075321623fb90ebb181fe79ec3e23dcb081ef15906
   ```

步骤 2：创建待签字符串

对HTTP请求进行规范并取得请求的哈希值后，将其与签名算法、签名时间一起组成待签名字符串。

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

伪代码中参数说明如下。

- **Algorithm**

  签名算法。对于SHA 256，算法为SDK-HMAC-SHA256。

- **RequestDateTime**

  请求时间戳。与请求消息头X-Sdk-Date的值相同，格式为YYYYMMDDTHHMMSSZ。

- **HashedCanonicalRequest**

  经过哈希处理的规范请求。

上述例子得到的待签字符串为：

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

步骤 3：计算签名

将APP secret和创建的待签字符串作为加密哈希函数的输入，计算签名，将二进制值转换为十六进制表示形式。

伪代码如下：

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

其中HMAC指密钥相关的哈希运算，HexEncode指转十六进制。伪代码中参数说明如表1-3所示。

表 1-3 参数说明

参数名称	参数解释
APP secret	签名密钥，认证用的ak和sk编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全
string to sign	创建的待签字符串

假设APP secret为12345678-1234-1234-1234-123456781234，则计算得到的signature为：

```
cb978df7c06ac242bab1d1b39d697ef7df4806664a6e09d5f5308a6b25043ea2
```

步骤 4：添加签名信息到请求头

在计算签名后，将它添加到Authorization的HTTP消息头。Authorization消息头未包含在已签名消息头中，主要用于身份验证。

伪代码如下：

```
Authorization header创建伪码：
Authorization: algorithm Access=APP key, SignedHeaders=SignedHeaders, Signature=signature
```

需要注意的是算法与Access之前没有逗号，但是SignedHeaders与Signature之前需要使用逗号隔开。

得到的签名消息头为：

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

得到签名消息头后，将其分别以Authorization和x-Authorization参数增加到原始HTTP请求头内容中，请求将被发送给API网关，由API网关完成身份认证。身份认证通过后，该请求才会发送给后端服务进行业务处理。