API 网关 SDK 使用指南

版本 V1.0（2020.12.17）

目录

一、使用须知

SDK 已经对加签验签逻辑做了封装，使用 SDK 可直接调用 API。

SDK 下载路径：找到对应的产品，点击产品 SDK-SDK 下载，选择对应版本的 SDK 即可进行下载。具体路径可参考下图示例。

1.1、类名称推导规则

请求响应对象类名称推导规则：

- URI 命名空间(首字母大写)+URI 自定义部分(首字母大写)+Request 或 Response；
- 拼接后的字符串中的点（.）、下划线（_）、中杠（-）、斜杠（/）的后一个字母大写并且（.）、下划线（_）、中杠（-）、斜杠（/）替换为空；

参考样例：

URI：/napi/v1/bt.cash/contractSignResultNotify

bt.cash：URI 命名空间

contractSignResultNotify：URI 自定义部分

请求对象类名称推导过程：

Bt.cashContractSignResultNotifyRequest==>Bt.CashContractSignResultNotifyRequest==>BtCashContractSignResultNotifyRequest

响应对象类名称推导过程：

Bt.cashContractSignResultNotifyResponse==>Bt.CashContractSignResultNotifyResponse==>BtCashC

ontractSignResultNotifyResponse

请求响应中的子对象以成员内部类的形式生成到请求响应对象中，避免多个 API 中子对象名重复。

1.2、SDK 文件目录

SDK 文件目录结构例如：

```
- jdd-open-demo
    * lib
        * jdd-open-sdk-1.5.1.jar     `sdk 的核心处理包，通讯、加解密、加验签等`
        * jdd-api-domain-1.0.0.jar    `商户请求响应实体类`
        * wyaks-security-1.0.9.jar    `JD 加解密 SDK`
        * bcpkix-jdk15on-1.59.jar    `开源密码包`
        * bcprov-jdk15on-1.59.jar    `开源密码包 `
    * src `demo 文件夹，各种接口调用示例`
        * com.jdd.open.demo.esu.hapi.HapiController          `hapi 示例代码`
        * com.jdd.open.demo.esu.napi.NapiDemoController      `napi 示例代码`
        * com.jdd.open.demo.esu.smapi.SmapiDemoInvoker      `smapi 示例代码`
        * com.jdd.open.demo.esu.spi.SpiDemoController        `spi 示例代码`

        * resources
            * config.application.yml `Spring Boot 项目启动配置文件`
            * config.log4j2.yml `Spring Boot 项目日志配置文件`
            * static.js.jquery.min.js    `hapi 表单提供所依赖的 js`
            * templates.messages.form.html   `hapi 表单提交的 HTML`
            * jdd_sdk_config.properties `配置文件，调用之前需确保已配置正确（包括环境地址、公私钥、
应用信息配置等）`
            * log4j2.xml     `Spring Boot 项目日志配置文件`
    * pom.xml
    * API 网关 SDK 使用指南 V1.0.pdf     `SDK 使用指南 PDF`
    * README.md `SDK 使用指南`
```

二、环境及配置

2.1、环境准备

JDK 版本：jdk1.7 及以上

2.2、SDK 配置信息

SDK 配置信息请参考文件目录中的 jdd_sdk_config.properties

使用 AKS 加解密配置如下：

说明：jdd.esu.appId、jdd.esu.openPublicKey、jdd.esu.appPrivateKey 根据实际情况修改，其他配置无特殊要求不需要调整。

```
#应用 ID，用户 ID 和应用 ID 一致，用户类型默认 0
jdd.esu.appId=4a5634fca025625860d8f3ad9c3b005f
#服务端密钥系统类型
jdd.esu.jddKeyType=AKS
#京东平台公钥
jdd.esu.openPublicKey=
jdd.esu.appKeyType=PAIR
#合作伙伴私钥
jdd.esu.appPrivateKey=
#客户端盐值 AKS 加密方式不需要
jdd.esu.md5Salt=
#加密算法，支持 3DES_RSA，无特殊要求不需要修改
jdd.esu.encryptType=3DES_RSA
#加签算法，支持 SHA256withRSA
jdd.esu.signType=SHA256withRSA
#服务端调用地址！测试：http://59.151.67.121:1058、生产：https://api.jddglobal.com
jdd.esu.server=http://59.151.67.121:1058
```

使用默认方式加解密配置如下：

说明：jdd.esu.appId、jdd.esu.openPublicKey、jdd.esu.appPrivateKey、jdd.esu.md5Salt 根据实际情况修改，其他配置无特殊要求不需要调整

```
#应用 ID，用户 ID 和应用 ID 一致，用户类型默认 0
jdd.esu.appId=6b232709d5a25d554b59ea05b4a621e0
#服务端密钥系统类型，秘钥类型 DEFAULT
jdd.esu.jddKeyType=DEFAULT
#京东平台公钥，
jdd.esu.openPublicKey=
jdd.esu.appKeyType=PAIR
#合作伙伴私钥
jdd.esu.appPrivateKey=
#客户端盐值，
jdd.esu.md5Salt=1uDc795BMT78aGPoBc9735v0
#加密算法，支持 NONE、RSA、ENV_RSA，无特殊要求不需要修改
jdd.esu.encryptType=ENV_RSA
#加签算法，支持 MD5_RSA、SHA1withRSA，无特殊要求不需要修改
jdd.esu.signType=SHA1withRSA
#服务端调用地址！测试：http://59.151.67.121:1058、生产：https://api.jddglobal.com
jdd.esu.server=http://59.151.67.121:1058
```

三、项目接入

3.1、普通项目接入（推荐使用）

1、将当前 SDK 中 lib 下所有 jar 文件添加至目标项目 lib 下(如果是 intellij idea 可以选中 lib 目录 右键选择 Add as Library)

2、将配置文件 jdd_sdk_config.properties 拷贝至目标项目 classpath 根目录

3、运行 src 下的示例代码（运行之前需要一些简单参数的组装）

3.2、maven 工程结构项目接入

1、仅将 lib 下的 jdd-open-sdk-1.5.1.jar、jdd-api-domain-1.0.0.jar、wyaks-security-1.0.9.jar、bcpkix-jdk15on-1.59.jar、bcprov-jdk15on-1.59.jar 包安装至本地仓库或者远程仓库

2、在 pom.xml 中添加以下依赖

````
```
    <dependency>
        <groupId>com.jddglobal.open</groupId>
        <artifactId>jddglobal_open_sdk</artifactId>
        <version>1.5.1</version>
    </dependency>
    <dependency>
        <groupId>com.jddglobal.open</groupId>
        <artifactId>jdd_api__domain</artifactId>
        <version>1.0.0</version>
     </dependency>
```
````

> ###注意###

- bcpkix-jdk15on 和 bcprov-jdk15on 依赖以当前版本为主，其他版本可能会有兼容性问题！

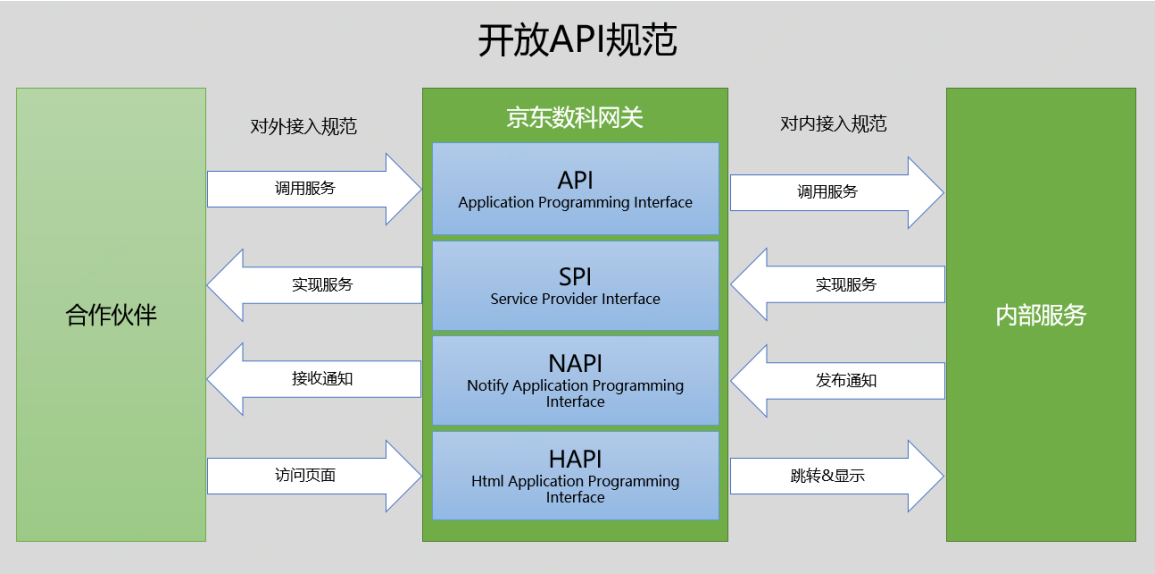
3、使用默认加载方式：将配置文件 jdd_sdk_config.properties 拷贝至目标项目 classpath 根目录下

四、调用说明

4.1、调用前准备

- 要提前在 jdd_sdk_config.properties 填写自己的用户参数
- 产品所有接口对应的 Request、Response 实体类都存在 lib 目录下的 jdd-api-domain.1.0.0.jar 中，包路径是 com.jddglobal.open.request、com.jddglobal.open.response

4.2、规范分类说明



规范名称	分类名称	分类编码	访问方向	说明
开放 API 规范	同步接口	API	合作伙伴 –> 京东数科	
	页面接口	HAPI	合作伙伴 –> 京东数科	
	反向接口	SPI	京东数科 –> 合作伙伴	
	通知接口	NAPI	京东数科 –> 合作伙伴	

4.3、调用方法及示例

Api 类型

调用方法:

注意: 接口编码类型为 SMAPI 和 MAPI 的接口都适用此方法，适用合作伙伴调用京东数科接口场景。
第一步：加载 jdd_sdk_config.properties 配置文件的信息，读取应用 ID、密钥、加密方式、加签方式...
JddClient jddClient = EsuSdkConfig.getJddClient();

第二步：组装请求参数
HttpBizDemoRequest request = new HttpBizDemoRequest();

request.setXxx("");

第三步：执行请求，请求的数据已经过加密加签发送给 JD 测，响应的数据已经过解密验签返回
HttpBizDemoResponse adaptResponse = jddClient.execute(request)

调用示例：

```
public class SmapiDemoInvoker {

    public static void main(String[] args) throws Exception {
        // 设置配置文件路径，默认为如下值，可不设置
        JddClient jddClient = EsuSdkConfig.getJddClient();
        HttpBizDemoRequest request = new HttpBizDemoRequest();
        request.setBal(123);

        HttpBizDemoResponse adaptResponse = jddClient.execute(request);
        System.out.println(JacksonUtils.toJson(adaptResponse));
    }
```

Spi 类型

调用方法：

注意：适用京东数科通知合作伙伴接口，并且需要合作伙伴返回结果的场景。

第一步：启动 Spring Boot 项目，JD 测访问 ip:端口/spi/demo

第二步：加载 jdd_sdk_config.properties 配置文件的信息，读取应用 ID、密钥、加密方式、加签方式...
DefaultSpiJddClient defaultSpiJddClient = new DefaultSpiJddClient(EsuSdkConfig.getAppInfo());

第三步：接受 JD 测发送过来的数据，函数内部已经做好对 JD 测密文数据进行解密验签工作
String bizContent = defaultSpiJddClient.receive(request);

第四步：商户对明文 bizContent 做业务逻辑处理

第五步：响应 JD 测信息，函数内部已做好对商户发送给 JD 测信息的加密加签工作
String body = defaultSpiJddClient.callback(request, response, spiResponse);
response.getWriter().write(body);

调用示例：

```
public class SpiDemoController {

    @PostMapping("/demo")
    public void demo(HttpServletRequest request, HttpServletResponse response) throws Exception {
```

```
        DefaultSpiJddClient defaultSpiJddClient = new
DefaultSpiJddClient(EsuSdkConfig.getAppInfo());
        String bizContent = defaultSpiJddClient.receive(request);
        log.info("demo-request,bizContent={}", bizContent);
        /**
         * TODO 如下是业务逻辑开始
         */
        Map<String, String> responseDemo = new HashMap<>();
        responseDemo.put("respDemo", bizContent);
        /**
         * 业务逻辑结束，开始封装响应
         */
        SpiResponse spiResponse = new SpiResponse("000000", "成功",
                DateUtils.formatDate(new Date(), "yyyyMMddHHmmss"),
JacksonUtils.toJson(responseDemo));
        String body = defaultSpiJddClient.callback(request, response, spiResponse);
        log.info("demo-response,body={}", body);
        //获得字符输出流
        response.setCharacterEncoding("utf-8");
        PrintWriter writer = response.getWriter();
        writer.write(body);
    }

}
```

Napi 类型

调用方法：

注意：适用合作伙伴通知商户，外部应用需明确返回 “success” 或者 “ok” 的应答，否则会进行 5 次分间隔重试，合作伙伴方接口必须保持幂等

第一步：启动 Spring Boot 项目，JD 测访问 ip:端口/napi/demo

第二步：加载 jdd_sdk_config.properties 配置文件的信息，读取应用 ID、密钥、加密方式、加签方式...
DefaultNapiJddClient defaultNapiJddClient = new DefaultNapiJddClient(EsuSdkConfig.getAppInfo());

第三步：接受 JD 测发送过来的数据，函数内部已经做好对 JD 测密文数据进行解密验签工作
OutResponse notifyContent = defaultNapiJddClient.notify(request);

第四步：商户对明文 bizContent 做业务逻辑处理

调用示例：

```
public class NapiDemoController {

    @PostMapping("/demo")
    public void demo(HttpServletRequest request, HttpServletResponse response) throws Exception {
        DefaultNapiJddClient defaultNapiJddClient = new
```

```
DefaultNapiJddClient(EsuSdkConfig.getAppInfo());
        OutResponse notifyContent = defaultNapiJddClient.notify(request);
        log.info("demo-request,notifyContent={}", notifyContent);
        /**
         * TODO 如下是业务逻辑
         */
        response.getWriter().write("OK");
    }
}
```

Hapi 类型

调用方法：

注意：适用合作伙伴调用京东数科 API，返回 H5 页面的场景。

第一步：启动 Spring Boot 项目，浏览器输入 ip:端口/auto，然后会跳转到 form.html 表单(demo 中是自动跳转)

第二步：form.html 表单会调用 ip:端口/getData 获取表单提交所需要的数据，进入到 com.jdd.open.demo.esu.hapi.HapiController.getData 方法中

第三步：加载 jdd_sdk_config.properties 配置文件的信息，读取应用 ID、密钥、加密方式、加签方式...
HapiJddClient hapiJddClient = new DefaultHapiJddClient(EsuSdkConfig.getAppInfo());

第四步：根据拼接业务请求 jsonParams 和请求时间 requestTime 获取 encrypt、sign、envKey
String[] args = hapiJddClient.gainEncryptAndSign(jsonParams, requestTime);

第五步：组装表单参数 result

第六步： form.html 表单提交

调用示例：

```
public class HapiController {

    @RequestMapping(value = "auto")
    public ModelAndView getFormIndex() {

        System.out.println("auto 自动跳转");
        return new ModelAndView("messages/form");
    }

    @RequestMapping(value = "getData")
    public Map<String, String> getData(HttpServletRequest request) throws Exception {
        Map<String, Object> data = new HashMap<>();
```

```
        SystemInfo systemInfo = new SystemInfo();
        systemInfo.setSysId("123");
        systemInfo.setToken("456");
        data.put("systemInfo",systemInfo);

        Map<String, String> param = new HashMap<>();
        param.put("clientAgent", "test agent");
        //00015   测试     00171   生产
        param.put("openApiParam", "{\"backUrl\":\"www.jd.com\"}");
        param.put("processId", "262");
        param.put("openSys", "A20200702110");
        param.put("openId", "YFKLXD001");
        param.put("clientIp", "100.100.100.100");
        param.put("openUser", "YFKLXD001");
        data.put("param",param);
        Map<String, Object> totalMap = new HashMap<>();
        totalMap.put("data", data);
        String jsonParams = JacksonUtils.toJson(totalMap);

        String requestTime = String.valueOf(System.currentTimeMillis());

        Map<String, String> result = new HashMap<>();

        HapiJddClient hapiJddClient = new DefaultHapiJddClient(EsuSdkConfig.getAppInfo());
        String[] args = hapiJddClient.gainEncryptAndSign(jsonParams, requestTime);
        result.put("encrypt", args[0]);
        result.put("sign", args[1]);
        result.put("envKey", args[2]);
        result.put("time", requestTime);
        result.put("userId", EsuSdkConfig.getAppInfo().getAppId());
        result.put("encryptType", EsuSdkConfig.getAppInfo().getEncryptType());
        result.put("signType", EsuSdkConfig.getAppInfo().getSignType());
        return result;
    }
}
```