JAVASDK 安全方案

V1.1

神州数码融信软件有限公司

2018 年 7 月

文 档 编 号		保 密 等 级	
作 者	开放平台项目组	最后修改日期	
审 核 人		最后审批日期	
批 准 人		最后批准日期	

修订记录

日期	版本	修订人	修订说明	审核人
2018-07-19	V1.0	卢培发	创建文档	
2018-07-20	V1.1	卢培发	修改了一些文字描述	

目　　录

1. 概述

1.1. 文档目的

本文档对 JAVASDK 的设计、功能实现原理进行说明，仅供合作商户参考。

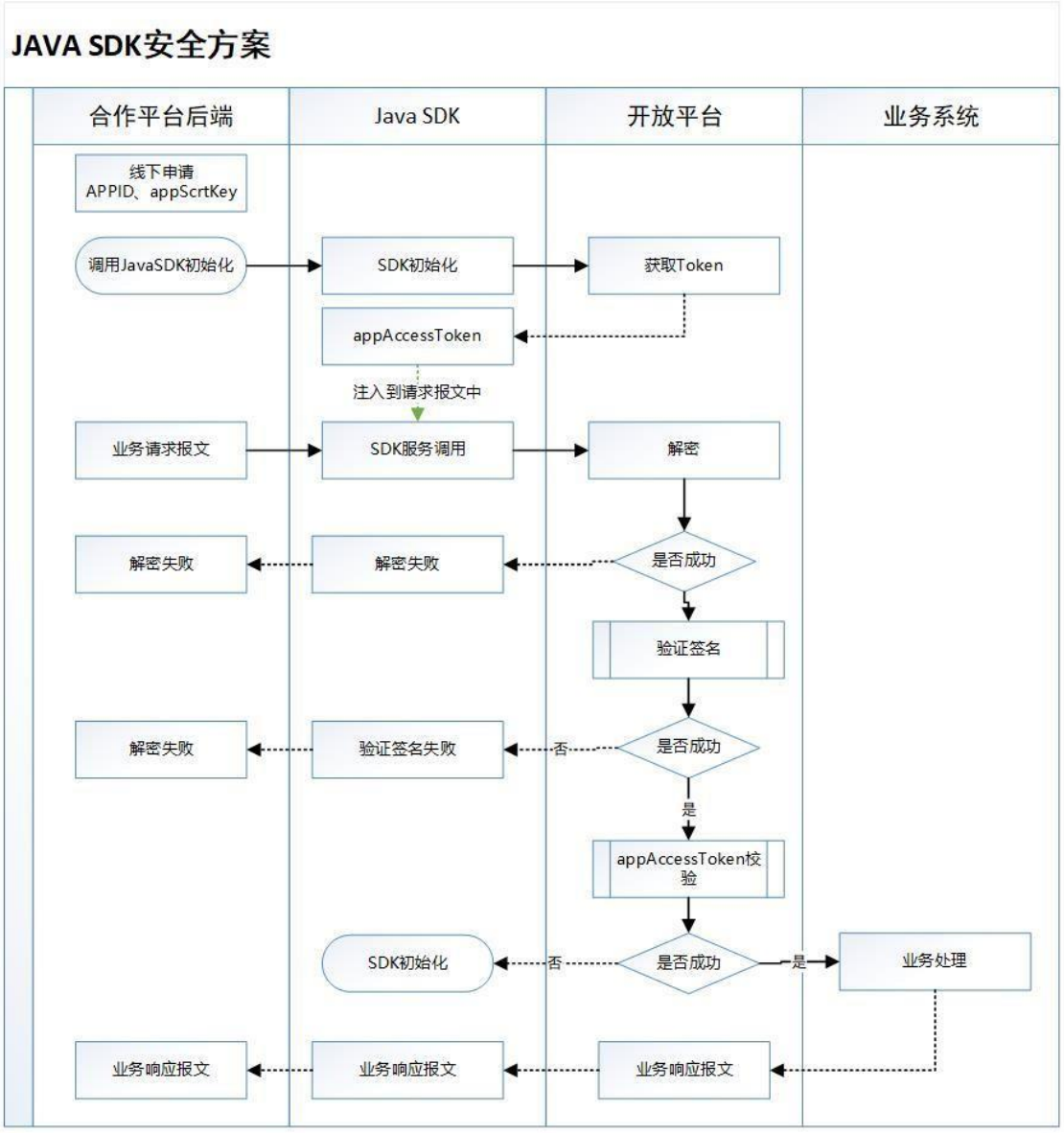
1.2. 适用范围

本文档的目标读者是合作商户的技术人员、测试人员。

2. 术语定义

缩写/术语	名词解释
appID	应用账号，商户调用javaSDK 之前必须首先先向开放平台申请一个 appID，appID 是开放平台给每个商户的唯一标识；
appScrtKey	应用密钥，这个同appID 一样，开放平台根据appID 给商户返回一个 appScrtKey；
appAccessToken	令牌，商户通过调用 javaSDK 中的 SDK 初始化 approveDev 方法，开放平台返回一个 appAccessToken 给商户，appAccessToken 令牌的有效期是 30 分钟，超时的话需要重新调用 SDK 初始化，进行获取新的 appAccessToken；
aeskey	随机秘钥，对加签后请求报文进行 AES 加密，把加签后的请求明文变成密文；
加密、解密	商户公钥对 aeskey 进行RSA 加密，商户用 aeskey 进行AES 解密；
加签、验签	商户私钥对请求报文进行加签，商户公钥对返回报文进行验签；
client.key	商户后台的私钥，由商户自己生成，用来对请求报文进行加签；
server.crt	由开放平台提供给商户的公钥，商户用这个公钥对 aeskey 进行加密；还要用这个公钥对返回报文进行验签。
AES	高级加密标准，一种对称秘钥加密算法。
RAS	RSA 加密算法，一种非对称加密算法。
SHA256	安全散列算法，SHA256 算法的哈希值大小为 256 位。

3. JAVASDK 安全方案



3.1. SDK 初始化

3.1.1. 功能说明

商户业务后台调用 javaSDK 初始化方法，向开放平台请求获取 **appAccessToken**。

3.1.2. 流程图

reqSign= RSA_sign（SHA256（reqData），私钥A）

AES（[reqData+reqSign]，aeskey）+RSA（aeskey，公钥 B）

respSign=RSA_sign（SHA256（respData），私钥B）

3.1.3. 流程说明

3.1.3.1 .请求 appAccessToken

商户系统后台调用 **javaSDK** 的初始化方法，方法入参中上传事先申请的 **appID** 和 **appScrtKey**，JavaSDK 对请求报文 reqData 进行加密加签，并将密文通过 HTTPS 的方式发往开发平台后端。

1、reqData=appID+appScrtKey+rdmNum

报文格式：**{**

“request”：{

“appID” : ”1b514715_6297_44be_b338_0f13985684f3”,

"appScrtKey":"c9fa0cdf-86eb-4845-b3a9-efb82a03322c",

"rdmNum":"h1qzj05m7bd79uqasy2gjie7lf2pzdtu"

}

}

报文要求：

（1）json 格式；

（2）红色部分的 appID 和 appScrKey 是必须由商户事先通过向开放平台申请获取，这两个字段是商户传进来的，rdmNum 由 javaSDK 内部自动生成；

（3）所有字段必须放在 request 里面。

2、Java SDK 对 reqData 进行排序，再进行 SHA256 哈希加密得到哈希值 hash；

3、用私钥A（client.key）对哈希值hash 进行加签，得到reqSign，并把reqSign 放入reqData 请求报文中；

加签报文：**{**

“request”:{

“appID” : ”1b514715_6297_44be_b338_0f13985684f3”,

"appScrtKey":"c9fa0cdf-86eb-4845-b3a9-efb82a03322c",

"rdmNum":"h1qzj05m7bd79uqasy2gjie7lf2pzdtu",

“sign”:”Hgajdj3848HHS8438==4hshgNFHghdjah==”

}

}

注：**sign** 也是放在 **request** 里面。

4、用随机秘钥 aeskey 对请求报文[reqData+reqSign]进行 AES 加密；

注：**aeskey** 是**javaSDK** 生成。

5、用公钥 B 对 aeskey 进行 RSA 加密；

6、商户后台把**AES**（**[reqData+ RSA_sign**（**SHA256**（**reqData**），私钥**A**）**]**，**aeskey**）**+RSA**（**aeskey**，公钥 **B**） 发给开放平台。

请求报文：**{**

（密文） **“request”**：**”SJHFSDJKCN238JSJDBDLXsjsdncjshSJDIADJIFCJU84g84==”**

“aeskey”:”SHJHSJH757FJDJHdjfhdjfk++++++====”

}

注：**aeskey** 是放在 **request** 里面。

3.1.3.2 .返回 appAccessToken

javaSDK 对响应报文 respData 进行解密验签，验签通过后，获取响应报文中的 **appAccessToken** 和**appID**；这两个值会存在 javaSDK 中，当商户后台调用服务接口的时候，javaSDK 会把这两个值注入请求报文中。

开放平台返回的响应报文：

AES（**[respData+RSA_sign**（**SHA256**（**respData**），私钥 **B**）**]**，**aeskey**）

1、用aeskey 解密响应密文，得到 respData+respSign；

注：**aeskey** 是请求部分生成的。

2、对解密出来的响应明文 respData 进行排序和 SHA256 哈希加密，并使用公钥 B 进行验签；

3、验签通过，获取 appAccessToken 和appID。

响应报文：

```
{    “mag” : ”验证成功”,
     “result”: {
                    “appID” : ”1b514715_6297_44be_b338_0f13985684f3”,
                    "appAccessToken":" c9fa0cdf-86eb-4845-b3a9-efb82a03322c "
          },
     “code” : “000000”
}
```

报文格式：

（1） **json** 格式；

（**2**）响应报文包括 **appID** 和 **appAccessToken**，并在 **result** 里面。

3.2. SDK 服务调用

3.2.1. 功能说明

SDK 服务调用：主要是商户后台向开放平台调用服务接口。

3.2.2. 流程图

reqSign= SHA256（reqData+appAccessToken+appID）

respSign= SHA256（respData）

3.2.3. 流程说明

3.2.3.1 .发送请求

商户后台调用服务接口，**javaSDK** 对请求报文 reqData 进行加密加签，并把密文通过 HTTPS 的方式发往开发平台。

1、reqData=业务数据+rdmNum；

报文格式：

```
{       “request”：{
                        "appAcceassToken":"c9fa0cdf-86eb-4845-b3a9-efb82a03322c",
                        “appID” : ”1b514715_6297_44be_b338_0f13985684f3”,

                         业务字段
                        "rdmNum":"h1qzj05m7bd79uqasy2gjie7lf2pzdtu"
                }
}
```

以 身份证 **OCR** 接口 为例：

```
{       “request”：{
                        "appAcceassToken":"c9fa0cdf-86eb-4845-b3a9-efb82a03322c",
                        “appID” : ”1b514715_6297_44be_b338_0f13985684f3”,

                        //业务字段
                        “TxnSrlNo”:”2018071848984772”
                        “IdentCrdPht”:”ifidjGDHSKhKJS29834JCKLDJJWEH”,
                        “BusTp”:”001”,
                        ……..
                        "rdmNum":"h1qzj05m7bd79uqasy2gjie7lf2pzdtu"
                }
}
```

报文要求：（**1**）**json** 格式；

（1） **SDK** 初始化得到的 **appAccessToken** 和初始化前申请的 **appID** 必须要在报文里面传值，这两个值 **javaSDK** 会自动添加；

（**3**）字段必须在 **request** 里面。

2、对[reqData+appAccessToken+appID]进行排序，然后 SHA256 哈希加密得到 hash；

3、reqSign=hash，并把 reqSign 放入 reqData 请求报文中；

加签报文：

{ request":{ "appAcceassToken":"c9fa0cdf-86eb-4845-b3a9-efb82a03322c",

"appID":"1b514715_6297_44be_b338_0f13985684f3",

业务字段

"rdmNum":"h1qzj05m7bd79uqasy2gjie7lf2pzdtu",

"sign":"Hgajdj3848HHS8438==4hshgNFHghdjah=="

}

}

注：**sign** 也是放在 **request** 里面。

4、用随机秘钥 aeskey 对请求报文 reqData+ appAccessToken +reqSign 进行AES256 加密；

注：**aeskey** 是**javaSDK** 生成。

5、用公钥 B 对 aeskey 进行 RSA 加密；

6、商户后台把 **AES**（**[reqData+ SHA256**（**reqData**）**]**，**aeskey**）**+RSA**（**aeskey**，公钥 **B**）发给开放平台。

请求报文：**{**

（密文） **"request"："SJHFSDJKCN238JSJDBDLXsjsdncjshSJDIADJIFCJU84g84=="**

"aeskey":"SHJHSJH757FJDJHdjfhdjfk++++++====

}

注：**aeskey** 是不放在 **request** 里面的。

3.2.3.2 .返回响应

javaSDK 对响应报文 respData（密文）进行解密验签，解密成功并验签通过后，获取响应报文中 **respData**（明文），返回给商户。

开放平台返回的响应报文：

AES（**[respData+ SHA256**（**respData**）**]**，**aeskey**）

1、用aeskey 解密响应密文，得到 respData+respSign；

注：**aeskey** 是请求部分生成的。

2、对解密出来的响应明文 respData 进行排序和 SHA256 哈希加密，得到的哈希值跟 respSign 进行比较，进行验签；

3、验签通过，得到响应报文 respData。

返回报文（以上面 身份证 **OCR** 接口 为例）：

```
{    “msg” : ”SUCCESS”,
     “result”: {
                        “Adr” : ”广东省”,
                        "Birthday":"1993-03-18",
                        “GndTp”:”男”,
                        ……..
                },
     “code” :“000000”
}
```

报文格式：

（**1**）**json** 格式；

（**2**）响应报文在 **result** 里面。