数据共享平台 SDK 接口设计

（JAVA 版）

版本：V 1.0.1

2017 年 12 月

文档修订记录

版本编号	*变化状态	变更内容和范围	变更日期	变更人	批准日期	批准人
1.0.0	A	创建数据共享平台 SDK 接口设计	2017/11/28	宋嘉		
1.0.1	M	修改上链模版数据 jwt 封装方式	2017/12/14	宋嘉		

*变化状态：A——增加，M——修改，D——删除，N——正式发布

目录

1 数据共享平台 SDK 相关说明

1.1 sdk 引用

将附件“open-datashare-sdk-1.0.1-RELEASE-with-dependencies.jar”包导入应用程序工程内，并确保工程中已导入 sdk 相关依赖 jar，依赖列表如下：

依赖包名称	版本
commons-lang3	3.5
fastjson	1.2.28
okhttp	3.4.1
slf4j-api	1.7.25

Maven 方式引用配置：

```
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.25</version>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.7.25</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.5</version>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.28</version>
</dependency>
<dependency>
    <groupId>com.squareup.okhttp3</groupId>
    <artifactId>okhttp</artifactId>
    <version>3.4.1</version>
</dependency>
```

1.2 jdk 要求

open-datashare-sdk-1.0.1-RELEASE-with-dependencies.jar
是基于 jdk1.7 开发与编译，故引用该包的工程环境必须为 jdk1.7+。

1.3 配置文件说明

- sdk 需要使用到两个配置文件：

  1、 通用参数配置文件：imi-config.properties

参数名称	描述	示例
通信地址		
url.share.addData.prepare	数据上链参数预处理接口地址	由我司提供
url.share.addData.commit	数据上链交易提交接口地址	
url.share.getData	上链数据查询接口地址	
url.share.verifyData	上链数据验证接口地址	
url.share.query.transactionInfo	链上交易信息查询接口地址	
url.share.query.blockInfo	链上块信息查询接口地址	
基本信息		
imi.vportId	数字身份号	现暂由我司分配，后期通过网络申请
imi.name	数字身份名称	由用户自定义（如：第三方系统名称）

  2、 区块链私钥存储 KeyStore 文件：imi-ks

  KeyStore 文件暂由我司提供，后期通过网络申请；

- 配置文件存储位置：

  1、 应用程序内部存储：

  存储于应用程序 classpath 根目录下，文件路径为

  imi/imi-config.properties

  imi/imi-ks

  2、 应用程序外部存储

  存储于系统文件目录下，如 linux 下的存储位置如下：

  /home/imi/imi-config.properties

  /home/imi/imi-ks

  在应用程序启动是通过调用 sdk 的参数初始化配置接口“IMIConfiguration.initConfigPath(String imiConfigPath, String imiKsPath, String imiKsPass)”传入两个配置文件路径以及 keystore 解密密码（暂由我司提供，后期通过网络申请，用户自定义）。

2 接口设计

接口相关内容可参考 API 文档包 open-datashare-sdk-doc-1.0.0

2.1 实体定义

2.1.1 StoreDataParams

名称	上链数据存储接口参数对象		StoreDataParams
字段名	**类型（是否必填）**	**说明**	**备注**
data	String(Y)	需要上链的用户原数据	JSON 格式的模版数据，SDK 中会对原数据进行 jwt 封装
typeCode	String(Y)	上链数据类型编号	JSON 模版编号
subjectCid	String(Y)	上链数据所属用户的身份证号	
subjectName	String(Y)	上链数据所属用户的身份证姓名	
issuerVportId	String(Y)	上链数据发布者数字身份编号	发布者即第三方业务系统
isAppend	int(N)	上链数据方式	添加：0，更新：1，默认是 0

2.1.2 GetDataParams

名称	上链数据查询接口参数对象		GetDataParams
字段名	**类型（是否必填）**	**说明**	**备注**
typeCode	String(Y)	上链数据类型编号	JSON 模版编号
subjectCid	String(Y)	上链数据所属用户的身份证号	
subjectName	String(Y)	上链数据所属用户的身份证姓名	
issuerVportId	String(Y)	上链数据发布者数字身份编号	发布者即第三方业务系统
endTime	String(N)	按上链时间查询：截止时间	默认当前时间，格式：yyyy-MM-dd hh:mm:ss
num	Integer(N)	查询数据条数	默认值：10
transactionHash	String(N)	交易编号可以查询到唯一的上链数据	该条件与上述条件互斥，该条件有效时，其他查询条件失效

2.1.3 VerifyDataParams

名称	上链数据验证接口参数对象		VerifyDataParams
字段名	**类型（是否必填）**	**说明**	**备注**
transaction	String(N)	交易编号可以查询到唯一的	

Hash		上链数据	

2.1.4 QueryTransactionInfoParams

名称	链上交易信息查询接口参数对象		QueryTransactionInfoParams
字段名	**类型 （是否必填）**	**说明**	**备注**
transaction Hash	String(N)	交易 Hash 值	

2.1.5 QueryBlockInfoParams

名称	上链数据验证接口参数对象		QueryBlockInfoParams
字段名	**类型 （是否必填）**	**说明**	**备注**
type	String(N)	块查询参数类型	固定三种类型： “blockhash”、 “blocknumber”、 “transactionhash”
value	String(N)	块查询参数值	

2.1.6 GetDataVO

名称	上链数据查询接口响应结果对象		GetDataVO
字段名	**类型 （是否必填）**	**说明**	**备注**
data	String(Y)	已上链的用户数据	JSON 格式的模版数据，SDK 中会对接口返回的 jwt 数据进行解封
typeCode	String(Y)	上链数据类型编号	JSON 模版编号
transaction Hash	String(Y)	上链数据的区块链交易编号	
createTime	String(Y)	数据上链时间	格式：yyyy-MM-dd hh:mm:ss

2.1.7 VerifyDataVO

名称	上链数据验证接口响应结果对象		VerifyDataVO
字段名	**类型 （是否必填）**	**说明**	**备注**
data	String(Y)	已上链的用户数据	JSON 格式的模版数据，SDK 中会对接口返回的 jwt 数据进行解封

typeCode	String(Y)	上链数据类型编号	JSON 模版编号
transactionHash	String(Y)	上链数据的区块链交易编号	

2.1.8 TransactionInfoVO

名称	链上交易信息查询接口响应结果对象		TransactionInfoVO
字段名	**类型（是否必填）**	**说明**	**备注**
hash	String(Y)	交易 Hash 值	
nonce	BigInteger(Y)	交易所属用户的交易序号	
blockHash	String(Y)	交易所在块的块 Hash 值	
blockNumber	BigInteger(Y)	交易所在块的块编号	
transactionIndex	BigInteger(Y)	交易索引号	
from	String(Y)	交易发送者地址	
to	String(Y)	交易接收者地址	
value	BigInteger(Y)	交易发送的代币数量	
gasPrice	BigInteger(Y)	燃料价格	
gas	BigInteger(Y)	燃料数量	
input	String(Y)	交易的输入信息	
r	String(Y)	用来生成交易签名	
s	String(Y)	用来生成交易签名	
v	int(Y)	用来生成交易签名	

2.1.9 BlockInfoVO

名称	链上块信息查询接口响应结果对象		BlockInfoVO
字段名	**类型（是否必填）**	**说明**	**备注**
number	BigInteger(Y)	区块编号	
hash	String(Y)	区块的 Hash 值	
parentHash	String(N)	父节点 Hash	
nonce	BigInteger(Y)	工作量证明机制计算所得的答案	
sha3Uncles	String(Y)	叔叔节点数据的 SHA3 编码	
logsBloom	String(Y)	存储 log 的数据结构	
transactionsRoot	String(Y)	交易树的树根	
stateRoot	String(Y)	状态树的树根 Hash	
receiptsRoot	String(Y)	当前区块，所有交易收条默克尔树根 Hash	
miner	String(Y)	挖矿挖到该区块的帐户	
mixHash	String(Y)	该值与 nonce 一起证	

		明挖矿工作量	
difficulty	BigInteger(Y)	区块挖矿难度	
totalDifficulty	BigInteger(Y)	挖到该区块时，区块链挖矿总体难度	
extraData	String(Y)	额外信息	
size	BigInteger(Y)	区块大小	
gasLimit	BigInteger(Y)	燃料消耗上限	
gasUsed	BigInteger(Y)	该区块中交易已消耗的燃料	
timestamp	BigInteger(Y)	时间戳	
transactions	List<TransactionInfoVO>(N)	块上的交易信息列表	
uncles	List<String>(N)	叔叔节点	

2.2 接口定义

2.2.1 配置文件路径初始化接口

接口名称	配置文件路径初始化接口				
类名	**方法名**				
IMIConfiguration	initConfigPath(String imiConfigPath, String imiKsPath, String imiKsPass)				
功能	此接口调用需要放在应用程序启动时调用				
输入参数			**输出参数 result**		
imiConfigPath	String(Y)	Sdk 通用参数配置文件路径			
imiKsPath	String(Y)	私钥 KeyStore 文件路径			
imiKsPass	String(Y)	私钥 KeyStore 解密密码			

示例:

```
// 配置文件系统存储绝对路径
String imiConfigPath = "/home/imi/imi-config.properties";
String imiKsPath = "/home/imi/imi-ks";
String imiKsPass =
"8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92";
try {
    IMIConfiguration.initConfigPath(imiConfigPath, imiKsPath , imiKsPass);
} catch (FileNotFoundException e) {
    e.printStackTrace();
}
```

2.2.2 数据上链存储接口

接口名称		数据上链存储接口			
类名		方法名			
DataSharingRouter		storeData(StoreDataParams params)			
功能		此接口内封装了数据共享平台数据上链的两个 Http 接口调用，以及交易数据本地签名操作			
输入参数			输出参数 result		
params	StoreDataParams (Y)	上链数据存储参数	transactionHash	String	数据上链交易 Hash 值

示例:

```
// 上链原始数据
String rawData = "{\"claim\":{\"type\":\"中华人民共和国居民身份证
\",\"assertion\":{\"name\":\"向南\",\"sex\":\"男\",\"ethnicity\":\"汉族
\",\"address\":\"北京朝阳酒仙桥\",\"cin\":\"43072119801024008x\",\"doi\":\"2015-02-
15\",\"doe\":\"2035-02-15\",\"authority\":\"北京市公安局
\",\"image\":{\"type\":\"image/Jpeg\",\"data\":/9j/4AAQSkZJRgABAQEAYABgAAD/2wBDAAgG
BgcGBQgHBwcJCQgKDBQNDAsLDBkSEwKDcpLDAxNDQ0HyWY4dh6GilyRfQd2f/9k=}},\"recipient\":{\
"vportId\":\"0x3f182ebdbfec6af397f59ab1d125ff3e90dc6318\"}}}";

// 数据模版类型编号
String typeCode = "10100000004";

// 上链数据所属用户身份证号
String subjectCid = "43072119801024008x";

// 上链数据所属用户身份证姓名
String subjectName = "向南";

// 上链模版数据签发信息添加
JSONObject rawDataJson = JSON.parseObject(rawData);
JSONObject issuer = new JSONObject();
issuer.put("name", CommonConstants.imiName);
issuer.put("hashed", false);
issuer.put("publicKey", KSConstant.publicKey);
issuer.put("vportId", CommonConstants.imiVportId);
rawDataJson.put("issuer", issuer);
rawDataJson.put("type", "CitizenIdentityClaim");
rawDataJson.put("iat", System.nanoTime());
rawDataJson.put("charset", "utf-8");
rawDataJson.put("@context", "http://www.blockcerts.org/schema/1.2/context.json");

// 第三方业务系统自行实现分布式锁
Lease lock = null;
```

```
try {
    StoreDataParams params = new StoreDataParams();
    params.setData(rawDataJson.toJSONString());
    params.setTypeCode(typeCode);
    params.setSubjectCid(subjectCid);
    params.setSubjectName(subjectName);
    params.setIssuerVportId(CommonConstants.imiVportId); // 第三方业务系统数字身份
号
    params.setIsAppend(0); // 0:添加数据，1:更新数据

    // 加锁
    lock = Lock.getLock(hander.LOCK_DATASHARE_STOREDATA);

    String transactionHash = DataSharingRouter.storeData(params);

    System.out.printf("storeData 响应：transactionHash=[%s]\n", transactionHash);
} catch (IMIRpcException e) {
    e.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
} finally {
    // 解锁
    if (lock != null) {
        Lock.unLock(lock);
    }
}
```

2.2.3 上链数据查询接口

接口名称		上链数据查询接口			
类名		**方法名**			
DataSharingRouter		getData(GetDataParams params)			
功能		此接口可以通过多查询条件查询多条上链数据集合，也可以通过上链数据交易 Hash 值查询单条数据			
输入参数			**输出参数 result**		
params	GetDataParams (Y)	上链数据查询参数	result	List<GetDataVO>	上链数据查询结果

示例：

```
/****************** 通过多存储条件查询多数据集合 ******************/
// 数据模版类型编号
String typeCode = "10100000004";

// 上链数据所属用户身份证号
```

```
String subjectCid = "43072119801024008x";

// 上链数据所属用户身份证姓名
String subjectName = "向南";

// 查询上链数据条件：截止时间
String endTime = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss").format(new Date());

// 查询上链数据条件：数据条数
int num = 10;

try {
    GetDataParams params = new GetDataParams();
    params.setTypeCode(typeCode);
    params.setSubjectCid(subjectCid);
    params.setSubjectName(subjectName);
    params.setIssuerVportId(CommonConstants.imiVportId); // 第三方业务系统数字身份
号(来自配置文件)
    params.setEndTime(endTime);
    params.setNum(num);

    List<GetDataVO> dataList = DataSharingRouter.getData(params);
    System.out.printf("getData 响应：dataList=[%s]\n",
JSONObject.toJSONString(dataList));
} catch (IMIRpcException e) {
    e.printStackTrace();
}


/****************** 通过交易 Hash 值查询单条数据 ******************/
// 数据上链的交易 Hash 值
String transactionHash =
"0xc60ec052a6fe245ff9a361755814080c8196aecbdc0e218da064b8739ece71fe";

try {
    GetDataParams params = new GetDataParams();
    params.setTransactionHash(transactionHash);

    List<GetDataVO> dataList = DataSharingRouter.getData(params);
    GetDataVO vo = null;
    if (null != dataList && !dataList.isEmpty()) {
        vo = dataList.get(0);
    }
    System.out.printf("getData 响应：data=[%s]\n", JSONObject.toJSONString(vo));
} catch (IMIRpcException e) {
```

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

2.2.4 上链数据验证接口

接口名称		上链数据验证接口			
类名		方法名			
DataSharingRouter		verifyData(VerifyDataParams params)			
功能		此接口可以通过上链数据查询接口获得的上链数据中的交易 Hash 值查询改数据的验证数据			
输入参数			输出参数 result		
params	VerifyDataParams(Y)	上链数据验证参数	result	VerifyDataVO	上链数据验证数据查询结果

示例:

```
// 上链数据的交易 Hash 值
String transactionHash =
"0xc60ec052a6fe245ff9a361755814080c8196aecbdc0e218da064b8739ece71fe";

try {
    VerifyDataParams params = new VerifyDataParams();
    params.setTransactionHash(transactionHash);

    VerifyDataVO vo = DataSharingRouter.verifyData(params);

    System.out.printf("verifyData 响应: data=[%s]\n", JSONObject.toJSONString(vo));
} catch (IMIRpcException e) {
    e.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
}
```

2.2.5 链上交易信息查询接口

接口名称	链上交易信息查询接口
类名	方法名
DataSharingRouter	queryTransactionInfo(QueryTransactionInfoParams params)
功能	此接口可以通过交易 Hash 值查询区块链上的交易信息
输入参数	输出参数 result

params	QueryTransactionInfoParams(Y)	链上交易信息查询参数	result	TransactionInfoVO	链上交易信息查询结果

示例:

```
// 通过交易 Hash 值查询交易信息
String transactionHash =
"0xc60ec052a6fe245ff9a361755814080c8196aecbdc0e218da064b8739ece71fe";

try {
    QueryTransactionInfoParams params = new QueryTransactionInfoParams();
    params.setTransactionHash(transactionHash);

    TransactionInfoVO info = DataSharingRouter.queryTransactionInfo(params);
    System.out.printf("testQueryTransactionInfo =>> info: [ %s ]\n",
JSON.toJSONString(info));

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

2.2.6 链上块信息查询接口

接口名称		链上块信息查询接口			
类名		**方法名**			
DataSharingRouter		queryBlockInfo(QueryBlockInfoParams params)			
功能		此接口可以通过交易 Hash 值、块 Hash 值、块编号查询区块链上的块信息			
输入参数			**输出参数 result**		
params	QueryBlockInfoParams(Y)	链上块信息查询参数	result	BlockInfoVO	链上块信息查询结果

示例:

```
// 块查询条件
String transactionHash =
"0xb1c9a80c0c8782e80a169c2a4cb382e356de5bbba95e1d9113d9b72c462e3dfa";
String blockHash =
"0x5caf54c240d5dc397fb4f9a46ef11976b85170f96cd4a516d9a98152600c1945";
long blockNumber = 8665;
```

```
try {
    // 方式一：通过块 Hash 值查询块信息
    QueryBlockInfoParams params = new QueryBlockInfoParams();
    params.setType("blockhash");
    params.setValue(blockHash);

    BlockInfoVO info = DataSharingRouter.queryBlockInfo(params);

    System.out.printf("testQueryBlockInfo-blockHash =>> info: [ %s ]\n",
JSON.toJSONString(info));

    // 方式二：通过块编号查询块信息
    params = new QueryBlockInfoParams();
    params.setType("blocknumber");
    params.setValue(String.valueOf(blockNumber));

    info = DataSharingRouter.queryBlockInfo(params);

    System.out.printf("testQueryBlockInfo-blockNumber =>> info: [ %s ]\n",
JSON.toJSONString(info));

    // 方式三：通过交易 Hash 值查询块信息
    params = new QueryBlockInfoParams();
    params.setType("transactionhash");
    params.setValue(transactionHash);

    info = DataSharingRouter.queryBlockInfo(params);
    long end2 = System.currentTimeMillis();

    System.out.printf("testQueryBlockInfo-transactionHash =>> info: [ %s ]\n",
JSON.toJSONString(info));

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