weid-sample Documentation

发布 *version*

weidentity team

2019 年 07 月 30 日

Contents:

1 环境准备 3

2 快速体验 5

3 命令行方式使用 7

4 spring-boot 服务方式使用 13

weid-sample 是基于 WeIdentity 开发的 Java 应用样例程序，提供了一整套的流程演示，可以帮您快速理解 WeIdentity 的运行机制，您也可以参考该样例程序，开发您的 Java 应用。

CHAPTER 1

环境准备

您可以参考 WeIdentity JAVA SDK 安装部署文档中的准备工作章节进行准备。

CHAPTER 2

快速体验

我们提供了两种方式体验 weid-sample：

- 命令行方式使用 （推荐方式）
- spring-boot 方式使用

CHAPTER 3

命令行方式使用

3.1 整体介绍

命令行方式比较完整的模拟了各个 WeIdentity 角色的工作流程，可以帮您快速体验 WeIdentity 也业务流程和运行机制。各个角色的基本流程如下：

- Issuer

  创建 WeID

  注册成为 Authority Issuer

  注册 CPT

  创建 Credential

- User Agent

  创建 WeID

  通过 AMOP 获取 Verifier 发布的 Presentation Policy

  创建 Presentation

  打包 Presentation 成 QRcode 或者 Json 串，发送给 Verifier

- Verifier

  获取 User Agent 的 Presentation

  验证 Presentation

3.1.1 1. 配置与部署

1.1 下载 weid-sample 源码:

```
git clone https://github.com/WeBankFinTech/weid-sample.git
```

1.2 部署 weid-java-sdk 与配置基本信息

- 安装部署 weid-java-sdk

  weid-sample 需要依赖 weid-java-sdk，您需要参考WeIdentity JAVA SDK 安装部署完成 weid-java-sdk 的安装部署，并参照Java 应用集成章节完成 weid-sample 的配置。

- 配置 Committee Member 私钥

注解: 此项配置并非必要。由于注册 Authority Issuer 需要委员会机构成员（Committee Member ）权限，若您不是发布智能合约的机构，您无需关注此配置项。若您是智能合约发布的机构，您可以参考以下进行配置:

将您在部署 WeIdentity 智能合约阶段生成的私钥文件拷贝至 `weid-sample/keys/priv/` 目录中，此私钥后续将用于注册 Authority Issuer，weid-sample 会自动加载。

- 修改节点和机构配置

多个角色之间会使用 AMOP 进行通信，根据 AMOP 协议，每个机构需要配置为连接不同的区块链节点。

```
cd weid-sample
vim src/main/resources/weidentity.properties
```

关键配置如下:

`blockchain.orgid` : 机构名称。样例以 organizationA 为例，请修改为 organizationA。

`nodes` : 区块链节点信息。你可以修改为您区块链网络中的任一节点即可。

配置样例:

```
blockchain.orgid=organizationA
nodes=10.10.10.10:20200
```

1.3 User Agent 服务配置

- 编译 weid-sample

如果您是第一次运行 weid-sample，您需要先进行编译:

```
chmod +x *.sh
./build.sh
```

- 启动 AMOP 服务

weid-sample 里的 AMOP 服务是模拟 Verifier 向 User Agent 发送获取秘钥的请求，因此 Verifier 和 User Agent 需要连接同一条链中的不同的区块链节点。先启动 Verifier 进程：

```
./command.sh daemon
```

运行成功，会启动 Verifier 的 AMOP 服务，输出如下日志：

```
the AMOP server start success.
```

- 修改 User Agent 配置

在启动完 Verifier 进程之后，还需要修改 User Agent 的配置，确保 User Agent 连接的区块链节点和 Verifier 连接的区块链节点在同一条链上，且连接的是不同的区块链节点：

```
vim dist/conf/weidentity.properties
```

此处主要是修改机构名称和区块链节点配置，要确保和 Verifier 连接的不是同一个区块链节点。

配置样例：

```
blockchain.orgid=organizationB
nodes=10.10.10.11:20200
```

3.1.2 2. 基本流程的演示

- Issuer 操作流程演示

```
./command.sh issuer
```

若运行成功，则会打印包括创建 WeID、注册成为 Authority Issuer、注册 CPT 和创建 Credential 等运行流程。

以下为截取的部分流程日志：

```
--------- start issuer ----------
issuer() init...

begin to createWeId...
```

(下页继续)

(续上页)

```
createWeId result:

result:(com.webank.weid.protocol.response.CreateWeIdDataResult)
weId: did:weid:1:0x7a276b294ecf0eb7b917765f308f024af2c99a38
userWeIdPublicKey:(com.webank.weid.protocol.base.WeIdPublicKey)
    publicKey:␣
↪14431083876897147338218517164635545928469555951949020873197753983829667965157417451
    95118210554711531306779199915498227256788151940687396693589185508570578 4
userWeIdPrivateKey:(com.webank.weid.protocol.base.WeIdPrivateKey)
    privateKey:␣
↪46686865859949148045125507514815998920467147178097685958028816903332430030079
errorCode: 0
errorMessage: success
transactionInfo:(com.webank.weid.protocol.response.TransactionInfo)
blockNumber: 2098
transactionHash: 0x20fc5c2730e4636248b121d31ffdbf7fa12e95185068fc1dea060d1afa9d554e
transactionIndex: 0

begin to setPublicKey...

setPublicKey result:

result: true
errorCode: 0
errorMessage: success
transactionInfo:(com.webank.weid.protocol.response.TransactionInfo)
blockNumber: 2099
transactionHash: 0x498d2bfd2d8ffa297af699c788e80de1bd51c255a7365307624637ae5a42f3a1
transactionIndex: 0
```

- User Agent 操作流程演示

```
./command.sh user_agent
```

运行成功，则会打印包括创建 WeID、通过 AMOP 获取 Verifier 发布的 Presentation Policy、创建 Presentation 以及打包 Presentation 成 QRcode 或者 Json 串的流程。以下为截取的部分日志：

```
--------- start User Agent ----------
userAgent() init...

begin to create weId for useragent...
```

(下页继续)

(续上页)

```
createWeId result:

result:(com.webank.weid.protocol.response.CreateWeIdDataResult)
weId: did:weid:1:0x38198689923961e8ecd6d57d88d027b1a6d1daf2
userWeIdPublicKey:(com.webank.weid.protocol.base.WeIdPublicKey)
    publicKey:␣
↪12409513077193959265896252693672990701614851618753940603742819290794422690048786166
    77748624449230242365328258533877448834753636236821653645295685212386945 6
userWeIdPrivateKey:(com.webank.weid.protocol.base.WeIdPrivateKey)
    privateKey:␣
↪11700070604387246310492373601720779844791990854359896181912833510050901695117
errorCode: 0
errorMessage: success
transactionInfo:(com.webank.weid.protocol.response.TransactionInfo)
blockNumber: 2107
transactionHash: 0x2474141b82c367d8d5770a7f4d124aeaf985e7fa3e3e2f7f98eeed3d38d862f5
transactionIndex: 0
```

- Verifier 操作流程演示

```
./command.sh verifier
```

运行成功，则会打印 Verifier 反序列化 Presentation 以及验证 Presentation 的过程。以下为截取的部分日志，详细流程可以参考代码实现：

```
--------- start verifier ----------
verifier() init...

begin get the presentation json...
```

至此，您已经体验了 weid-sample 实现的各个角色的运行流程，实现的入口在 `com.webank.weid.demo.server.SampleApp`，您可以参考进行您的 Java 应用开发。

CHAPTER 4

spring-boot 服务方式使用

4.1 整体介绍

使用 spring-boot 方式，weid-sample 程序将作为一个后台进程运行，您可以使用 http 方式体验交互流程。

4.1.1 1. 下载 weid-sample 源码：

```
git clone https://github.com/WeBankFinTech/weid-sample.git
```

4.1.2 2. 配置与部署

2.1 下载 weid-sample 源码：

```
git clone https://github.com/WeBankFinTech/weid-sample.git
```

2.2 部署 weid-java-sdk 与配置基本信息

- 安装部署 weid-java-sdk

weid-sample 需要依赖 weid-java-sdk，您需要参考WeIdentity JAVA SDK 安装部署完成 weid-java-sdk 的安装部署，并参照Java 应用集成章节完成 weid-sample 的配置。

- 配置 Committee Member 私钥

  将您在部署 WeIdentity 智能合约阶段生成的私钥文件拷贝至 `weid-sample/keys/priv/` 目录中，此私钥后续将用于注册 Authority Issuer，weid-sample 会自动加载。

注解: 此项配置并非必要，注册 Authority Issuer 需要委员会机构成员（Committee Member ）权限，发布智能合约时生成的公私钥对会自动成为委员会机构成员，若您不是发布智能合约的机构，您无需关注此配置项。若您是智能合约发布的机构，您可以参考以下进行配置:

2.3 基本流程的演示

2.3.1 编译和运行

- 编译 weid-sample

```
cd weid-sample
chmod +x *.sh
./build.sh
```

- 启动 weid-sample 服务:

```
./start.sh
```

若启动成功，则会打印以下信息:

```
[main] INFO  AnnotationMBeanExporter() - Registering beans for JMX exposure on startup
[main] INFO  Http11NioProtocol() - Initializing ProtocolHandler ["https-jsse-nio-20190"]
[main] INFO  Http11NioProtocol() - Starting ProtocolHandler ["https-jsse-nio-20190"]
[main] INFO  NioSelectorPool() - Using a shared selector for servlet write/read
[main] INFO  Http11NioProtocol() - Initializing ProtocolHandler ["http-nio-20191"]
[main] INFO  NioSelectorPool() - Using a shared selector for servlet write/read
[main] INFO  Http11NioProtocol() - Starting ProtocolHandler ["http-nio-20191"]
[main] INFO  TomcatEmbeddedServletContainer() - Tomcat started on port(s): 20190 (https)␣
↪20191 (http)
[main] INFO  SampleApp() - Started SampleApp in 3.588 seconds (JVM running for 4.294)
```

2.3.2 流程演示

以下将为您演示假设您的服务部署在本地，地址是 `127.0.0.1`，服务端口是 `20191`。

- 创建 WeID

```
curl -l -H "Content-type: application/json" -X POST   http://127.0.0.1:20191/createWeId
```

若调用成功，则会打印以下信息：

```
{
    "result":{
        "weId":"did:weid:101:0xd613fbc0249f2ce5088ed484fa6b7b51ecb95e24",
        "userWeIdPublicKey":{
            "publicKey":
↪"31709029240872128509950537062055120804451989634302874297218468255989889984667160405337824673421192065
↪"
        },
        "userWeIdPrivateKey":null
    },
    "errorCode":0,
    "errorMessage":"success",
    "transactionInfo":{
        "blockNumber":60643,
        "transactionHash":
↪"0xc73b7ba6af39614761423dc8fcbbbc7e5f24c82e8187bc467cf0398b4ce4330b",
        "transactionIndex":0
    }
}
```

表明创建的 WeID 是 did:weid:101:0xd613fbc0249f2ce5088ed484fa6b7b51ecb95e24。

- 注册 Authority Issuer

```
curl -l -H "Content-type: application/json" -X POST -d '{"issuer":
↪"did:weid:101:0xd613fbc0249f2ce5088ed484fa6b7b51ecb95e24","org-id":"webank"}'
http://127.0.0.1:20191/registerAuthorityIssuer
```

运行成功，则会打印以下信息：

```
{
    "result":true,
    "errorCode":0,
    "errorMessage":"success",
    "transactionInfo":{
        "blockNumber":60668,
        "transactionHash":
↪"0xa0b84473705da2679cfec9119e2cdef03175df0f1af676e0579d5809e4e8d6cd",
```

(下页继续)

(续上页)

```
        "transactionIndex":0
    }
}
```

- 注册 CPT

运行成功，则会打印以下信息：

```
{
    "result":{
        "cptId":1189,
        "cptVersion":1
    },
    "errorCode":0,
    "errorMessage":"success",
    "transactionInfo":{
        "blockNumber":60676,
        "transactionHash":
↪"0x72d55eb1d020acd09b115177a46e230ffdb0177ab5dd74e16765d79338522093",
        "transactionIndex":0
    }
}
```

表明注册 CPT 成功，CPT ID 为 1189。

- 创建 Credential

创建 Credential 依赖于具体的 CPT，参数里的 cptId 传入刚刚注册的 CPT 的 ID：

若运行成功，则会打印以下信息：

```
{
    "result":{
        "credential":{
            "context":"https://github.com/WeBankFinTech/WeIdentity/blob/master/context/v1
↪",
            "id":"e4f4accd-6026-4fd0-9392-1379ddd4f778",
            "cptId":1189,
            "issuer":"did:weid:101:0xd613fbc0249f2ce5088ed484fa6b7b51ecb95e24",
            "issuanceDate":1564371227764,
            "expirationDate":1595475227763,
            "claim":{
                "gender":"F",
```

(下页继续)

(续上页)


```
                "name":"zhangsan",
                "id":"did:weid:101:0xf36fb2308d36bb94c579f568bdf670743d949deb"
            },
            "proof":{
                "creator":"did:weid:101:0xd613fbc0249f2ce5088ed484fa6b7b51ecb95e24",
                "signature":"G2kD4u4jrnmYbq/
↪oVl9idmTEQzP3a0KEomHGJaVpWzhITIE+dDYSRMyF9TDy+jPANpYRJGg7pGnANM+QeJ9Ba00=",
                "created":"1564371227764",
                "type":"EcdsaSignature"
            },
            "signature":"G2kD4u4jrnmYbq/
↪oVl9idmTEQzP3a0KEomHGJaVpWzhITIE+dDYSRMyF9TDy+jPANpYRJGg7pGnANM+QeJ9Ba00=",
            "proofType":"EcdsaSignature"
        },
        "disclosure":{
            "name":1,
            "id":1,
            "gender":1
        }
    },
    "errorCode":0,
    "errorMessage":"success",
    "transactionInfo":null
}
```


表明创建 Credential 成功，Credential 的具体信息为输出中的 Credential 字段对应的内容。

- 验证 Credential

若运行成功，则会打印以下信息：


```
{
    "result":true,
    "errorCode":0,
    "errorMessage":"success",
    "transactionInfo":null
}
```


表明 Credential 验证成功。

至此，您已经体验了 weid-sample 实现的各个角色的运行流程，实现的入口在 `com.webank.weid.demo.server.SampleApp`，您可以参考进行您的 Java 应用开发。