环境准备

系统环境准备

目前测试环境为：

- Ubuntu 16.04 / MacOS 10.15
- JDK 1.8+

启动链

进入到FISCO BCOS链运行所在目录（以下该目录简称为`$fisco`），执行如下命令：

1. 通过如下命令下载用于准备链环境的脚本：

   `curl -LO https://github.com/FISCO-BCOS/FISCO-BCOS/releases/download/v2.3.0/build_chain.sh && chmod u+x build_chain.sh`

2. 搭建单群组4节点联盟链：

   ```
   ./build_chain.sh -l "127.0.0.1:4" -p 30300,20200,8545
   ```

   执行成功后会在`nodes`子目录下生成所有相关文件

3. 启动链

   ```
   nodes/127.0.0.1/start_all.sh
   ```

4. 成功启动后可以通过以下命令查看以启动的节点进程：

   ```
   ps -ef | grep -v grep | grep fisco-bcos
   ```

准备控制台

进入`$fisco`目录并执行以下操作：

1. 通过如下命令获取控制台：

```
nodes/127.0.0.1/download_console.sh
```

2. 通过如下命令拷贝控制台配置文件：

```
cp -n console/conf/applicationContext-sample.xml
console/conf/applicationContext.xml
```

3. 配置控制台证书：

```
cp nodes/127.0.0.1/sdk/* console/conf/
```

4. 至此完成控制台的启动工作，通过如下命令启动控制台：

```
console/start.sh
```

5. 在命令行中输入如下命令以查询当前块高：

```
getBlockNumber
```

由于在FISCO BCOS中只有新交易到达才会增长块高，因此这里返回的结果为`0`

拷贝合约

1. 进入到smart-audit源码文件下，然后进入到`src/fisco`子目录，通过如下命令拷贝所有合约文件到控制台所维护的目录下：

```
cp -r audit/*.sol audit/interface
$fisco/console/contracts/solidity/
```

2. 通过如下命令拷贝人脸识别预言机服务相关的智能合约到控制台所维护的目录下：

```
cp oracles/face/*.sol
$fisco/console/contracts/solidity/
```

3. 通过如下命令拷贝物体识别预言机服务相关的智能合约到控制台所维护的目录下：

```
cp oracles/identify/*.sol
$fisco/console/contracts/solidity/
```

4. 通过如下命令拷贝时间服务预言机服务相关的智能合约到控制台所维护的目录下：

```
cp oracles/time/*.sol
$fisco/console/contracts/solidity/
```

5. 通过如下命令拷贝定位服务预言机服务相关的智能合约到控制台所维护的目录下：

```
cp oracles/location/*.sol
$fisco/console/contracts/solidity/
```

合约拷贝完成后即可在控制台命令行中通过合约名找到相应的智能合约，后续所有操作都将在控制台命令行中执行

合约调用

部署合约

部署时间合约

1. 通过如下命令部署合约

```
deployByCNS DummyTimeService 1.0
```

如果部署合约成功会得到如下响应消息：

```
contract address:
0x7edac4c1fd59d55130806dbe537b718594189374
```

2. 可以通过如下命令查询时间服务的部署合约：

```
queryCNS DummyTimeService
```

若步骤1部署成功会得到如下响应：

```
-----------------------------------------------------------
---------------------------------------
|                    version                    |
          address                      |
|                      1.0                      |
0x7edac4c1fd59d55130806dbe537b718594189374  |
-----------------------------------------------------------
---------------------------------------
```

可以看到这里查询到的合约地址与步骤1的返回结果一致

3. 通过如下命令可以对时间合约的验证方法进行测试：

```
callByCNS DummyTimeService:1.0 validate 0 [""]
```

目前验证方法有90%的概率通过验证，如果成功会返回相应的交易hash，响应消息如下：

```
transaction hash:
0xb27a1be3394c382bbafbbaea951799292b4e6e64e484e4f0eda68
030ffe5b808
```

如果验证失败则返回如下响应消息：

```
时间超出正常工作范围
```

部署定位合约

1. 通过如下命令部署合约

```
deployByCNS DummyLocationService 1.0
```

如果部署合约成功会得到如下响应消息：

```
contract address:
0x66038422a9700ca5af804e1bb53a77a20e806de2
```

2. 可以通过如下命令查询时间服务的部署合约：

```
queryCNS DummyTimeService
```

若步骤1部署成功会得到如下响应：

```
----------------------------------------------------------
---------------------------------------
|                    version                    |
          address                    |
|                      1.0                      |
0x66038422a9700ca5af804e1bb53a77a20e806de2  |
----------------------------------------------------------
---------------------------------------
```

可以看到这里查询到的合约地址与步骤1的返回结果一致

3. 通过如下命令可以对时间合约的验证方法进行测试：

```
callByCNS DummyLocationService:1.0 validate 0 [""]
```

目前验证方法有90%的概率通过验证，如果成功会返回相应的交易hash，响应消息如下：

```
transaction hash:
0x76bb610b0384cef67325c66ea1b26cd482cc1932ffe5f0cd804b9
3ecc6515e0e
```

如果验证失败则返回如下响应消息：

```
地理位置超出正常工作范围
```

部署人脸识别合约

1. 通过如下命令部署合约

   ```
   deployByCNS DummyFaceService 1.0
   ```

   如果部署合约成功会得到如下响应消息：

   ```
   contract address:
   0x943eb039fe84f35e7fb8c09535f46441449a76c9
   ```

2. 可以通过如下命令查询时间服务的部署合约：

   ```
   queryCNS DummyFaceService
   ```

   若步骤1部署成功会得到如下响应：

   ```
   ------------------------------------------------------------
   ---------------------------------------
   |                   version                    |
            address                     |
   |                     1.0                      |
   0x943eb039fe84f35e7fb8c09535f46441449a76c9  |
   ------------------------------------------------------------
   ---------------------------------------
   ```

   可以看到这里查询到的合约地址与步骤1的返回结果一致

3. 通过如下命令可以对时间合约的验证方法进行测试：

   ```
   callByCNS DummyFaceService:1.0 validate 0 [""]
   ```

   目前验证方法有>80%的概率通过验证，如果成功会返回相应的交易hash，响应消息如下：

```
transaction hash:
0xe61d943eead331b9dba26e8ea6fecc00f5dc0f6e026b1ccb5a175
b3209066b06
```

如果验证失败则返回如下响应消息：

```
非本人操作
```

部署物体识别合约

1. 通过如下命令部署合约

```
deployByCNS DummyIdentifyService 1.0
```

如果部署合约成功会得到如下响应消息：

```
contract address:
0xa348d87bb437b88dacd551be17e6111ec0b1f745
```

2. 可以通过如下命令查询时间服务的部署合约：

```
queryCNS DummyIdentifyService
```

若步骤1部署成功会得到如下响应：

```
--------------------------------------------------------
---------------------------------------
|                   version                    |
          address                      |
|                     1.0                      |
0xa348d87bb437b88dacd551be17e6111ec0b1f745  |
--------------------------------------------------------
---------------------------------------
```

可以看到这里查询到的合约地址与步骤1的返回结果一致

3. 通过如下命令可以对时间合约的验证方法进行测试：

```
callByCNS DummyIdentifyService:1.0 validate 0 [""]
```

目前验证方法有>80%的概率通过验证，如果成功会返回相应的交易hash，响应消息如下：

```
transaction hash:
0x3a72e3d2cd7476d35767ff89cc01566f118ee9e343f27bc99532f345e9b9ed81
```

如果验证失败则返回如下响应消息：

```
缺少需要识别物体的数据
```

部署审计业务合约

1. 通过如下命令部署审计合约

```
deployByCNS Audit 1.0 ["ZhangSan","LiSi"]
"0x943eb039fe84f35e7fb8c09535f46441449a76c9"
"0xa348d87bb437b88dacd551be17e6111ec0b1f745"
"0x7edac4c1fd59d55130806dbe537b718594189374"
"0x66038422a9700ca5af804e1bb53a77a20e806de2"
```

如果部署合约成功会得到如下响应消息：

```
contract address:
0x071a9de132bf0b24aa9c75aadc7d42baa151ec4c
```

2. 可以通过如下命令查询时间服务的部署合约：

```
queryCNS Audit
```

若步骤1部署成功会得到如下响应：

```
-------------------------------------------------------------
-----------------------------------
|                   version                    |
          address                    |
|                     1.0                      |
0x071a9de132bf0b24aa9c75aadc7d42baa151ec4c  |
-------------------------------------------------------------
-----------------------------------
```

可以看到这里查询到的合约地址与步骤1的返回结果一致

调用合约

合约维护人员查询

1. 通过如下命令查询合约维护人员

```
callByCNS Audit:1.0 getMaintainers
```

如果查询成功，会返回如下响应信息：

```
transaction hash:
0x7626fde6681d48fb73b0cba393ce92cbd2d2f48765fc1b1a3df37
0586b631e1b
-------------------------------------------------------------
-----------------------------------
Output
function: getMaintainers()
return type: (uint32[])
return value: ([0, 1])
-------------------------------------------------------------
-----------------------------------
```

这里返回的索引为注册审计部署合约填入的合约维护人员对应的ID值

录入审计当事人

1. 通过如下命令执行审计当事人录入操作：

```
callByCNS Audit:1.0 registerAuditee "WangWu"
```

如果注册成功，会返回如下响应消息，其中`registerAuditeeEvent`为注册成功后触发的事件：

```
transaction hash:
0xf5f62f562fb3f5e4de9f82230f00725e15a64d83bad74f96d368a
48e93d85aec
-------------------------------------------------------
---------------------------------------
Event logs
event signature: registerAuditeeEvent(uint32) index: 0
event value: (0)
-------------------------------------------------------
---------------------------------------
```

2. 根据返回的ID值按如下方式查询审计当事人信息：

```
callByCNS Audit:1.0 getAuditee 0
```

若调用成功会返回如下响应消息：

```
transaction hash:
0xc2361efe8f121ae06f842484e05c99d73fce2585c32732b4acc05
3a4e1cfc4aa
-------------------------------------------------------
---------------------------------------
Output
function: getAuditee()
return type: (string)
return value: (WangWu)
-------------------------------------------------------
---------------------------------------
```

录入规则

1. 通过如下命令执行规则录入操作：

```
callByCNS Audit:1.0 registerRules "AND", "Time", "(>=
9) AND (<= 18)", "Location", "IN(39.9 116.3 1000)",
"FaceRecognize", "", "ObjectRecognize", ""
```

如果注册成功，则会返回如下响应消息，其中`registerRuleEvent`为注册成功后触发的事件：

```
transaction hash:
0xcf48dd7ef0a48f9eb402d2a8d3576c5a7a982d6747209c59703fa
d65395f9af2
---------------------------------------------------------
---------------------------------------
Event logs
event signature: registerRuleEvent(uint32) index: 0
event value: (0)
---------------------------------------------------------
---------------------------------------
```

录入项目

1. 通过如下命令执行项目录入操作：

```
callByCNS Audit:1.0 registerProject "yucong_project" 0
0
```

如何注册成功，则会返回如下消息，其中`registerProjectEvent`为注册成功后触发的事件：

```
transaction hash:
0xcf48dd7ef0a48f9eb402d2a8d3576c5a7a982d6747209c59703fa
d65395f9af2
-------------------------------------------------------
---------------------------------------
Event logs
event signature: registerProjectEvent(uint32) index: 0
event value: (0)
-------------------------------------------------------
---------------------------------------
```

2. 根据返回的ID值按如下方式查询项目信息：

```
callByCNS Audit:1.0 getProject 0
```

若调用成功会返回如下响应消息：

```
transaction hash:
0xc2361efe8f121ae06f842484e05c99d73fce2585c32732b4acc05
3a4e1cfc4aa
-------------------------------------------------------
---------------------------------------
Output
function: getProject()
return type: (tuple(string,uint32,uint32))
return value: ([yucong_project, 0, 0])
-------------------------------------------------------
---------------------------------------
```

新增审计事件

1. 通过如下命令执行审计事件新增操作：

```
callByCNS Audit:1.0 addEvent 0 0 ["Time", "2020-06-
01T15:04:05.000Z", "Location", "39.901 116.299",
"FaceRecognize", "/9j/4SMF...", "ObjectRecognize",
"iVBORw0..."]
```

如果新增成功，则每一个预言机服务会打印相应的事件

2. 通过如下命令可查询到指定当事人在某个项目及规则下的所有事件：

```
callByCNS Audit:1.0 queryEvents 0 0
```

若调用成功会返回如下响应信息：

```
transaction hash:
0xda0ee52930ba0004ccbe1177b35ff52a8d0d79de496b656df70e7
6b0c7672cbb
-------------------------------------------------------
---------------------------------------
Output
function: queryEvents()
return type: (string[][])
return value: ([["Time", "2020-06-01T15:04:05.000Z",
"Location", "39.901 116.299", "FaceRecognize",
"/9j/4SMF...", "ObjectRecognize", "iVBORw0..."]])
-------------------------------------------------------
---------------------------------------
```

参考链接

1. FISCO BCOS官方节点搭建向导
2. 控制台命令官方文档说明