环境准备

系统环境准备

目前测试环境为：

- Ubuntu： 18.04、20.04 (注意XuperChain在MacOSX环境测试未通过，包括MacOSX上的docker)
- GCC：4.9+
- Golang：1.13.x （注意目前最新版本的1.14测试尚未支持）

然后依照XuperChain官方部署文档下载代码并进行编译，编译完成后进入到`output`子目录下。

启动链

1. 初始化链上配置：

   ```
   ./xchain-cli createChain
   ```

2. 启动链

   ```
   nohup ./xchain &
   ```

3. 成功启动后可以通过以下命令查看链上状态：

   ```
   ./xchain-cli status
   ```

   可在json类型的返回值中找到键为`height`，其值为当前链高

准备合约账户

1. 调用cli命令创建一个合约账户：

```
./xchain-cli account new --account 1111111111111111 --
fee 1000
```

这里指定账户名为`1111111111111111`，创建成功后相应的合约地址为`XC1111111111111111@xuper`，其中`xuper`为默认启动链时默认分配的链名称

2. 通过如下命令查询并验证新生成的合约账户ACL相关信息：

```
./xchain-cli acl query --account
XC1111111111111111@xuper
```

3. 准备符合权限的地址列表

   首先在`data`目录下创建子目录`acl`：

```
mkdir data/acl
```

   调用如下命令生成地址列表文件：

```
echo
"XC1111111111111111@xuper/dpzuVdosQrF2kmzumhVeFQZa1aYcd
gFpN" > data/acl/addrs
```

4. 向新生成的合约账户中转账

```
./xchain-cli transfer --to XC1111111111111111@xuper --
amount 100000000 --keys data/keys/
```

5. 成功转账后可以查询合约账户中的余额

```
./xchain-cli account balance XC1111111111111111@xuper
```

   这里会返回余额和步骤4中指定的`amount`相同

合约调用

进入到smart-audit源码文件下，然后进入到`src/xchain`子目录

编译合约

1. 运行如下命令进行wasm合约编译：

   ```
   make
   ```

   如果编译成功会在相同目录下生成`contract_audit.wasm`、`contract_identify.wasm`、`contract_time.wasm`、`contract_face.wasm`、`contract_location.wasm`五个文件

2. 通过如下命令将步骤1中生成的所有文件拷贝到xchain程序子目录下：

   ```
   cp *.wasm [your path]/data/blockchain/xuper/wasm/
   ```

   其中`[your path]`为上一节系统环境准备中所描述的`output`所在目录，至此不再使用smart-audit目录，可再次进入到上文中`output`目录下。

部署合约

部署时间合约

1. 通过如下命令进行试生成合约原始交易

   ```
   ./xchain-cli wasm deploy --account
   XC1111111111111111@xuper --cname Time -m -a '{}' -A
   data/acl/addrs -o time.output --keys data/keys --name
   xuper --runtime go
   data/blockchain/xuper/wasm/contract_time.wasm
   ```

   如果试生成成功会返回如下响应消息：

```
contract response:
The gas you cousume is: 5263887
You need add fee
```

2. 添加步骤1中的返回fee值后重复步骤1中的命令：

```
./xchain-cli wasm deploy --account
XC1111111111111111@xuper --cname Time -m -a '{}' -A
data/acl/addrs -o time.output --keys data/keys --name
xuper --runtime go
data/blockchain/xuper/wasm/contract_time.wasm --fee
5263887
```

若成功会返回如下响应消息，并在相同目录下生成名为`time.output`的文件：

```
contract response:
The gas you cousume is: 5263887
The fee you pay is: 5263887
```

3. 对步骤2中生成的原始交易签名

```
./xchain-cli multisig sign --tx time.output --output
time.sign --keys data/keys/
```

签名成功后会返回如下响应消息，并在相同目录下生成对原始交易的签名文件`time.sign`：

```
{
  "PublicKey": "{\"Curvname\":\"P-
256\",\"X\":74695617477160058757747208220371236837474210247114418775262229497812962582435,\"Y\":51348715319124770392993866417088542497927816017012182211244120852620959209571}",
  "Sign":
"MEUCIQDPokaKILO8SZnExeZwqosSpabdUXFi/fn6EYByt6aZxwIgClrHQbgb6/RUdjOEhOo67awiY8r+zMH/GVSxJpxR3E="
}
```

4. 将原始交易及签名发送到链上

```
./xchain-cli multisig send --tx time.output time.sign
time.sign
```

发送成功后会返回交易的Hash值如下：

```
Tx id:
0c47677b47b39e3c1b0c3d2c5a797ba833f54578685029555c9fc6bc6e04ff27
```

5. 查询合约账户验证部署结果

```
./xchain-cli account contracts --account
XC1111111111111111@xuper
```

当返回如下返回结果，包含刚才生成原始交易时指定的合约名则说明部署成功：

```
[
  {
    "contract_name": "Time",
    "txid":
"0c47677b47b39e3c1b0c3d2c5a797ba833f54578685029555c9fc6
bc6e04ff27",
    "desc":
"TWF5YmUgY29tbW9uIHRyYW5zZmVyIHRyYW5zYWN0aW9u",
    "timestamp": 1590651526456028700
  }
]
```


部署定位合约

1. 通过如下命令进行试生成合约原始交易

```
./xchain-cli wasm deploy --account
XC1111111111111111@xuper --cname Location -m -a '{}' -A
data/acl/addrs -o location.output --keys data/keys --
name xuper --runtime go
data/blockchain/xuper/wasm/contract_location.wasm
```

   如果试生成成功会返回如下响应消息：

```
contract response:
The gas you cousume is: 5279432
You need add fee
```

2. 添加步骤1中的返回fee值后重复步骤1中的命令：

```
./xchain-cli wasm deploy --account
XC1111111111111111@xuper --cname Location -m -a '{}' -A
data/acl/addrs -o location.output --keys data/keys --
name xuper --runtime go
data/blockchain/xuper/wasm/contract_location.wasm --fee
5279432
```

若成功会返回如下响应消息，并在相同目录下生成名为`location.output`的文件：

```
contract response:
The gas you cousume is: 5279432
The fee you pay is: 5279432
```

3. 对步骤2中生成的原始交易签名

```
./xchain-cli multisig sign --tx location.output --
output location.sign --keys data/keys/
```

签名成功后会返回如下响应消息，并在相同目录下生成对原始交易的签名文件`location.sign`：

```
{
  "PublicKey": "{\"Curvname\":\"P-
256\",\"X\":74695617477160058757747208220371236837474210247114418775262229497812962582435,\"Y\":51348715319124770392993866417088542497927816017012182211244120852620959209571}",
  "Sign":
"MEUCICw5cGtWIxTtrtwSNvMi03CFbBgfNqZ3S4iEy0t9OeZBAiEAzRr5188LmAdmT5X9KJySMIIcQg9awT1RxzE3fSxJ1kY="
}
```

4. 将原始交易及签名发送到链上

```
./xchain-cli multisig send --tx location.output
location.sign location.sign
```

发送成功后会返回交易的Hash值如下：

```
Tx id:
548fa01beab27affec7593cf4359b02b70ada5a5c45388cd1cb28419d6c0c3d3
```

5. 查询合约账户验证部署结果

```
./xchain-cli account contracts --account
XC1111111111111111@xuper
```

当返回如下返回结果，包含刚才生成原始交易时指定的合约名则说明部署成功：


```
[
  {
    "contract_name": "Location",
    "txid":
"548fa01beab27affec7593cf4359b02b70ada5a5c45388cd1cb284
19d6c0c3d3",
    "desc":
"TWF5YmUgY29tbW9uIHRyYW5zZmVyIHRyYW5zYWN0aW9u",
    "timestamp": 1590652021922963300
  },
  ...
]
```


部署人脸识别合约

1. 通过如下命令进行试生成合约原始交易

```
./xchain-cli wasm deploy --account
XC1111111111111111@xuper --cname FaceRecognize -m -a
'{}' -A data/acl/addrs -o face.output --keys data/keys
--name xuper --runtime go
data/blockchain/xuper/wasm/contract_face.wasm
```

如果试生成成功会返回如下响应消息：

```
contract response:
The gas you cousume is: 5263931
You need add fee
```

2. 添加步骤1中的返回fee值后重复步骤1中的命令：

```
./xchain-cli wasm deploy --account
XC1111111111111111@xuper --cname FaceRecognize -m -a
'{}' -A data/acl/addrs -o face.output --keys data/keys
--name xuper --runtime go
data/blockchain/xuper/wasm/contract_face.wasm --fee
5263931
```

若成功会返回如下响应消息，并在相同目录下生成名为`face.output`的文件：

```
contract response:
The gas you cousume is: 5263931
The fee you pay is: 5263931
```

3. 对步骤2中生成的原始交易签名

```
./xchain-cli multisig sign --tx face.output --output
face.sign --keys data/keys/
```

签名成功后会返回如下响应消息，并在相同目录下生成对原始交易的签名文件`face.sign`：

```
{
  "PublicKey": "{\"Curvname\":\"P-
256\",\"X\":74695617477160058757747208220371236837474210247114418775262229497812962582435,\"Y\":51348715319124770392993866417088542497927816017012182211244120852620959209571}",
  "Sign":
"MEYCIQDCoTq9oDH/yn7KePWGJ2nbYVW/uCZt5S3ETET78tAQFwIhAKa1B+9BrPKel87idutYOJn7+sHpEc3/l/t+YxIc8jy5"
}
```

4. 将原始交易及签名发送到链上

```
./xchain-cli multisig send --tx face.output face.sign
face.sign
```

发送成功后会返回交易的Hash值如下：

```
Tx id:
3956a358d16ab375b61001d972258af31c27a70987d5d7d972bdbe8
2295d36ad
```

5. 查询合约账户验证部署结果

```
./xchain-cli account contracts --account
XC1111111111111111@xuper
```

当返回如下返回结果，包含刚才生成原始交易时指定的合约名则说明部署成功：

```
[
  {
    "contract_name": "FaceRecognize",
    "txid":
"3956a358d16ab375b61001d972258af31c27a70987d5d7d972bdbe
82295d36ad",
    "desc":
"TWF5YmUgY29tbW9uIHRyYW5zZmVyIHRyYW5zYWN0aW9u",
    "timestamp": 1590652819298624100
  },
  ...
]
```

部署物体识别合约

1. 通过如下命令进行试生成合约原始交易

```
./xchain-cli wasm deploy --account
XC1111111111111111@xuper --cname ObjectRecognize -m -a
'{}' -A data/acl/addrs -o identify.output --keys
data/keys --name xuper --runtime go
data/blockchain/xuper/wasm/contract_identify.wasm
```

如果试生成成功会返回如下响应消息：

```
contract response:
The gas you cousume is: 5264740
You need add fee
```

2. 添加步骤1中的返回fee值后重复步骤1中的命令：

```
./xchain-cli wasm deploy --account
XC1111111111111111@xuper --cname ObjectRecognize -m -a
'{}' -A data/acl/addrs -o identify.output --keys
data/keys --name xuper --runtime go
data/blockchain/xuper/wasm/contract_identify.wasm --fee
5264740
```

若成功会返回如下响应消息，并在相同目录下生成名为`identify.output`的文件：

```
contract response:
The gas you cousume is: 5264740
The fee you pay is: 5264740
```

3. 对步骤2中生成的原始交易签名

```
./xchain-cli multisig sign --tx identify.output --
output identify.sign --keys data/keys/
```

签名成功后会返回如下响应消息，并在相同目录下生成对原始交易的签名文件`identify.sign`：

```
{
  "PublicKey": "{\"Curvname\":\"P-
256\",\"X\":74695617477160058757747208220371236837474210247114418775262229497812962582435,\"Y\":51348715319124770392993866417088542497927816017012182211244120852620959209571}",
  "Sign":
"MEUCIAdNsSg3z/70i5F7/1EyXMtCu22BvXPLZZPrMzt9SRC7AiEA9aDS6iOWbWcMRYgnhMQdVJ5RHOY3cnD1yOmRuYnLwgA="
}
```


4. 将原始交易及签名发送到链上

```
./xchain-cli multisig send --tx identify.output
identify.sign identify.sign
```

发送成功后会返回交易的Hash值如下：

```
Tx id:
a4b772681b08a31038e60abe903bdab8e1edf9a724a9d883081a5921541983c1
```

5. 查询合约账户验证部署结果

```
./xchain-cli account contracts --account
XC1111111111111111@xuper
```

当返回如下返回结果，包含刚才生成原始交易时指定的合约名则说明部署成功：

```
[
  {
    "contract_name": "ObjectRecognize",
    "txid":
"a4b772681b08a31038e60abe903bdab8e1edf9a724a9d883081a59
21541983c1",
    "desc":
"TWF5YmUgY29tbW9uIHRyYW5zZmVyIHRyYW5zYWN0aW9u",
    "timestamp": 15906532453798572OO
  },
  ...
]
```

部署审计业务合约

1. 通过如下命令进行试生成合约原始交易

```
./xchain-cli wasm deploy --account
XC1111111111111111@xuper --cname audit -m -a
'{"1":"bb","0":"aa"}' -A data/acl/addrs -o audit.output
--keys data/keys --name xuper --runtime go
data/blockchain/xuper/wasm/contract_audit.wasm
```

如果试生成成功会返回如下响应消息：

```
contract response:
The gas you cousume is: 5803059
You need add fee
```

2. 添加步骤1中的返回fee值后重复步骤1中的命令：

```
./xchain-cli wasm deploy --account
XC1111111111111111@xuper --cname audit -m -a '{}' -A
data/acl/addrs -o audit.output --keys data/keys --name
xuper --runtime go
data/blockchain/xuper/wasm/contract_audit.wasm --fee
5803059
```

若成功会返回如下响应消息，并在相同目录下生成名为`audit.output`的文件：

```
contract response:
The gas you cousume is: 5803059
The fee you pay is: 5803059
```

3. 对步骤2中生成的原始交易签名

```
./xchain-cli multisig sign --tx audit.output --output
audit.sign --keys data/keys/
```

签名成功后会返回如下响应消息，并在相同目录下生成对原始交易的签名文件`audit.sign`：

```
{
  "PublicKey": "{\"Curvname\":\"P-
256\",\"X\":74695617477160058757747208220371236837474210247114418775262229497812962582435,\"Y\":51348715319124770392993866417088542497927816017012182211244120852620959209571}",
  "Sign":
"MEQCIBTaWKboci8yk2scmijCrSbOAQ/tSXiOJCx7b5ggiICPAiBCeYTZXdcGaHjWVKKlOWRFLiqbLqr7vdfvYUuBXtPuAQ=="
}
```

4. 将原始交易及签名发送到链上

```
./xchain-cli multisig send --tx audit.output audit.sign
audit.sign
```

发送成功后会返回交易的Hash值如下：

```
Tx id:
bece8c1417e219e7327ee442439121bac957e6bc4e49c20bc38b95cbaefe55f9
```

5. 查询合约账户验证部署结果

```
./xchain-cli account contracts --account
XC1111111111111111@xuper
```

当返回如下返回结果，包含刚才生成原始交易时指定的合约名则说明部署成功：


```
[
  {
    "contract_name": "audit",
    "txid":
"bece8c1417e219e7327ee442439121bac957e6bc4e49c20bc38b95
cbaefe55f9",
    "desc":
"TWF5YmUgY29tbW9uIHRyYW5zZmVyIHRyYW5zYWN0aW9u",
    "timestamp": 1590655378727258000
  },
  ...
]
```


至此已完成所有的合约调用，以下为调用`./xchain-cli account contracts --account XC1111111111111111@xuper`返回的所有合约信息：


```
[
  {
    "contract_name": "audit",
    "txid":
"bece8c1417e219e7327ee442439121bac957e6bc4e49c20bc38b95cbae
fe55f9",
    "desc": "TWF5YmUgY29tbW9uIHRyYW5zZmVyIHRyYW5zYWN0aW9u",
    "timestamp": 1590655378727258000
  },
  {
    "contract_name": "face",
```

```
    "txid":
"3956a358d16ab375b61001d972258af31c27a70987d5d7d972bdbe82295d36ad",
    "desc": "TWF5YmUgY29tbW9uIHRyYW5zZmVyIHRyYW5zYWN0aW9u",
    "timestamp": 1590652819298624100
  },
  {
    "contract_name": "identify",
    "txid":
"a4b772681b08a31038e60abe903bdab8e1edf9a724a9d883081a5921541983c1",
    "desc": "TWF5YmUgY29tbW9uIHRyYW5zZmVyIHRyYW5zYWN0aW9u",
    "timestamp": 1590653245379857200
  },
  {
    "contract_name": "location",
    "txid":
"548fa01beab27affec7593cf4359b02b70ada5a5c45388cd1cb28419d6c0c3d3",
    "desc": "TWF5YmUgY29tbW9uIHRyYW5zZmVyIHRyYW5zYWN0aW9u",
    "timestamp": 1590652021922963300
  },
  {
    "contract_name": "time",
    "txid":
"0c47677b47b39e3c1b0c3d2c5a797ba833f54578685029555c9fc6bc6e04ff27",
    "desc": "TWF5YmUgY29tbW9uIHRyYW5zZmVyIHRyYW5zYWN0aW9u",
    "timestamp": 1590651526456028700
  }
]
```


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初始化是否成功：

```
./xchain-cli wasm invoke audit -a '{}' --method
getMaintainers -m
```

如果试生成成功，则会返回如下响应消息：

```
contract response: {"result":[{"Name":"aa","ID":0},
{"Name":"bb","ID":1}]}
The gas you cousume is: 100921
You need add fee
```

可以看到返回结果中包含了在部署审计合约时传入的运维人员，由于只是查询操作，我们不需要进一步将其发送到链上。

录入审计当事人

1. 通过如下命令注册一个审计当事人：

   ```
   ./xchain-cli wasm invoke audit -a '{"0":"ZhangSan"}' --
   method registerAuditee -m --output registerAuditee.out
   ```

   如果注册成功会返回如下响应消息：

   ```
   contract response: 0
   The gas you cousume is: 102281
   You need add fee
   ```

   注意contract response的返回值`0`为新注册的审计当事人对应的ID

2. 添加步骤2中的返回fee值后重复步骤2中的命令：

   ```
   ./xchain-cli wasm invoke audit -a '{"0":"ZhangSan"}' --
   method registerAuditee -m --output registerAuditee.out
   --fee 102281
   ```

   若生成成功则会在相同目录下生成名为`registerAuditee.out`的文件

3. 对步骤3中生成的原始交易签名

```
./xchain-cli multisig sign --tx registerAuditee.out --
output registerAuditee.sign --keys data/keys/
```

签名成功后在相同目录下生成对原始交易的签名文件`registerAuditee.sign`，并返回响应消息如下：

```
{
  "PublicKey": "{\"Curvname\":\"P-
256\",\"X\":74695617477160058757747208220371236837474210247114418775262229497812962582435,\"Y\":51348715319124770392993866417088542497927816017012182211244120852620959209571}",
  "Sign":
"MEYCIQDtNfxID18nLsJGvdAb3CSfnNzkLO+GraNp9zBOWQvosQIhALvjMyRoKqfjOKY6XscbQiRyUPJXh0qwRVOBip+RdJx8"
}
```

4. 将原始交易及签名发送到链上

```
./xchain-cli multisig send --tx registerAuditee.out
registerAuditee.sign registerAuditee.sign
```

发送成功后会返回交易的Hash值如下：

```
Tx id:
c2f19984b85cc256295940be05d00500a96274b47556481ceb26e23975ad87ef
```

5. 通过getAuditee接口查询以测试审计当事人是否注册成功

```
./xchain-cli wasm invoke audit -a '{"0":"0"}' --method
getAuditee -m
```

可以看到如下响应结果：

```
contract response: {"Name":"ZhangSan","ID":0}
The gas you cousume is: 101409
You need add fee
```

其中contract response所对应的值为查询得到的审计当事人，由于这里只是查询操作，因此不需要再附加fee进一步上链

录入规则

1. 通过如下命令注册一个规则：

```
./xchain-cli wasm invoke audit -a
'{"0":"AND","1":"Time","2":"(>= 9) AND (<=
18)","3":"Location","4":"IN(39.9 116.3
1000)","5":"FaceRecognize","6":"","7":"ObjectRecognize"
,"8":""}' --method registerRules -m --output
registerRules.out
```

如果注册成功会返回如下响应消息：

```
contract response: 0
The gas you cousume is: 506174
You need add fee
```

注意contract response的返回值`0`为新注册的规则对应的ID

2. 添加步骤2中的返回fee值后重复步骤2中的命令：

```
./xchain-cli wasm invoke audit -a
'{"0":"AND","1":"Time","2":"(>= 9) AND (<=
18)","3":"Location","4":"IN(39.9 116.3
1000)","5":"FaceRecognize","6":"","7":"ObjectRecognize"
,"8":""}' --method registerRules -m --output
registerRules.out --fee 506174
```

若生成成功则会在相同目录下生成名为`registerRules.out`的文件

3. 对步骤3中生成的原始交易签名

```
./xchain-cli multisig sign --tx registerRules.out --
output registerRules.sign --keys data/keys/
```

签名成功后在相同目录下生成对原始交易的签名文件`registerRules.sign`，并返回响应消息如下：

```
{
  "PublicKey": "{\"Curvname\":\"P-
256\",\"X\":74695617477160058757747208220371236837474210247114418775262229497812962582435,\"Y\":51348715319124770392993866417088542497927816017012182211244120852620959209571}",
  "Sign":
"MEUCIQD76TxvgTT6Yrr1cWp+h65BsKysY/AontFgJ4LI6Mt/5AIgKqZDso7pMFOYWwwATZtz46nf3W97nos89ZrkVUQLIto="
}
```

4. 将原始交易及签名发送到链上

```
./xchain-cli multisig send --tx registerRules.out
registerRules.sign registerRules.sign
```

发送成功后会返回交易的Hash值如下：

```
Tx id:
c1e56a85e594fa3752facf07ecc41a46870cae9ee9d950030c2ea1c4e3164c35
```

5. 通过getAuditee接口查询以测试审计当事人是否注册成功

```
./xchain-cli wasm invoke audit -a '{"0":"0"}' --method
getRules -m
```

可以看到如下响应结果：

```
contract response: {"Operator":"AND","Rules":
{"FaceRecognize":0,"Location":0,"ObjectRecognize":0,"Ti
me":0},"ID":0}"
The gas you cousume is: 101186
You need add fee
```

其中contract response所对应的值为查询得到的规则细节，由于这里只是查询操作，因此不需要再附加fee进一步上链

录入项目

1. 通过如下命令注册一个项目：

```
./xchain-cli wasm invoke audit -a '{"0":"POS
Audit","1":"This is a bank project, used by bank
employees to check if they did check the POS related
bussiness themselfs within the specified time and
location","2":"0","3":"0"}' --method registerProject -m
--output registerProject.out
```

如果注册成功会返回如下响应消息：

```
contract response: 0
The gas you cousume is: 103645
You need add fee
```

注意contract response的返回值`0`为新注册的规则对应的ID

2. 添加步骤2中的返回fee值后重复步骤2中的命令：

```
./xchain-cli wasm invoke audit -a '{"0":"POS
Audit","1":"This is a bank project, used by bank
employees to check if they did check the POS related
bussiness themselfs within the specified time and
location","2":"0","3":"0"}' --method registerProject -m
--output registerProject.out --fee 103645
```

若生成成功则会在相同目录下生成名为`registerProject.out`的文件

3. 对步骤3中生成的原始交易签名

```
./xchain-cli multisig sign --tx registerProject.out --
output registerProject.sign --keys data/keys/
```

签名成功后在相同目录下生成对原始交易的签名文件`registerProject.sign`，并返回响应消息如下：


```
{
  "PublicKey": "{\"Curvname\":\"P-
256\",\"X\":74695617477160058757747208220371236837474210247114418775262229497812962582435,\"Y\":51348715319124770392993866417088542497927816017012182211244120852620959209571}",
  "Sign":
"MEUCIQDPoO7c+2+BsX1zLCkMY3/6s+mnX+/W3TOJONAG+vmXFAIgIKWVj3H7Jzotd7q/PyAqfv1BD8yCoCwrawbiGxLwLdw="
}
```


4. 将原始交易及签名发送到链上

```
./xchain-cli multisig send --tx registerProject.out
registerProject.sign registerProject.sign
```

发送成功后会返回交易的Hash值如下：

```
Tx id:
b7048f38481087af18fa6ce84c72a90d57953ec937f35427a9326f34cad2d304
```

5. 通过getAuditee接口查询以测试审计当事人是否注册成功

```
./xchain-cli wasm invoke audit -a '{"0":"0"}' --method
getProject -m
```

可以看到如下响应结果：

```
contract response: {"Name":"POS
Audit","ID":0,"Description":"This is a bank project,
used by bank employees to check if they did check the
POS related bussiness themselfs within the specified
time and location","AuditeeRulesMap":{"
{"Name":"ZhanSan","ID":0}":"{"Operator":"AND","Rules":
{"FaceRecognize":0,"Location":0,"ObjectRecognize":0,"Ti
me":0},"ID":0}"}}
The gas you cousume is: 101858
You need add fee
```

其中contract response所对应的值为查询得到的规则细节，由于这里只是查询操作，因此不需要再附加fee进一步上链

新增审计事件

1. 通过如下命令新增一个审计事件：

```
./xchain-cli wasm invoke audit2 -a '{"a":"0", "b":"0",
"c":"1589532423", "d":"Time", "e":"2020-05-
29T13:04:05.000Z", "f":"Location", "g":"39.901
116.299", "h":"FaceRecognize", "i":"/9j/4SMF...",
"j":"ObjectRecognize", "k":"iVBORw0..."}' --method
addEvent -m --output addEvent.out
```

如果注册成功会返回如下响应消息：

```
contract response:
The gas you cousume is: 565194
You need add fee
```

注意contract response的返回值`0`为新注册的项目对应的ID

2. 添加步骤2中的返回fee值后重复步骤2中的命令：

```
./xchain-cli wasm invoke audit2 -a '{"a":"0", "b":"0",
"c":"1589532423", "d":"Time", "e":"2020-05-
29T13:04:05.000Z", "f":"Location", "g":"39.901
116.299", "h":"FaceRecognize", "i":"/9j/4SMF...",
"j":"ObjectRecognize", "k":"iVBORw0..."}' --method
addEvent -m --output addEvent.out --fee 565194
```

若生成成功则会在相同目录下生成名为`addEvent.out`的文件

3. 对步骤3中生成的原始交易签名

```
./xchain-cli multisig sign --tx addEvent.out --output
addEvent.sign --keys data/keys/
```

签名成功后在相同目录下生成对原始交易的签名文件`registerProject.sign`，并返回响应消息如下：


```
{
  "PublicKey": "{\"Curvname\":\"P-
256\",\"X\":74695617477160058757747208220371236837474210247114418775262229497812962582435,\"Y\":51348715319124770392993866417088542497927816017012182211244120852620959209571}",
  "Sign":
"MEQCIARr+1OpHr+EPsgGJJbz+2X62XujrRpltmkPMbx3vH2XAiBITC+v8sUHotVxYe7x8QAlZSsOqg33iqehBeXmm23pIw=="
}
```


4. 将原始交易及签名发送到链上

```
./xchain-cli multisig send --tx addEvent.out
addEvent.sign addEvent.sign
```

发送成功后会返回交易的Hash值如下：

```
Tx id:
19caa096ae9916355c99f853c6d3abe95d0a972a571164d92d51496
cddf995e1
```

5. 重复步骤1~4，再次新增事件
6. 通过queryEvents接口查询指定当事人在某个项目及规则下的所有事件

```
./xchain-cli wasm invoke audit -a '{"0":"0", "1":"0"}'
--method queryEvents -m
```

可以看到如下响应结果：


```
contract response: {"result":[{"ID":
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0],"Auditee":{"Name":"","ID":0},"Project":
{"Name":"","ID":0,"Description":""},"Rule":
{"Operator":"","Rules":
{},"ID":0},"Timestamp":1589532423,"Params":
["Time","2020-05-29T11:04:05.000Z","Location","39.901
116.299","FaceRecognize","/9j/4SMF...","ObjectRecognize
","iVBORw0..."],"Index":0},{"ID":
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0],"Auditee":{"Name":"","ID":0},"Project":
{"Name":"","ID":0,"Description":""},"Rule":
{"Operator":"","Rules":
{},"ID":0},"Timestamp":1589532423,"Params":
["Time","2020-05-29T13:04:05.000Z","Location","39.901
116.299","FaceRecognize","/9j/4SMF...","ObjectRecognize
","iVBORw0..."],"Index":1}]}
The gas you cousume is: 104000
You need add fee
```


其中contract response所对应的值为查询得到的所有事件细节，由于这里只是查询操作，因此不需要再附加fee进一步上链

参考链接

1. XuperChain环境部署
2. 创建合约
3. 超级链测试环境使用指南