基于Seele开发部署合约流程

基于Seele开发部署合约

编辑合约

你可以基于任何支持solidity的编辑器编写合约。下述将基于[remix在线编辑器]
((http://remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.4.24+commit.e67f0147.js&appVersion=0.7.7)
)为例，进行说明。

为了叙述方便，我们使用如下的简单合约示例：

```
pragma solidity ^0.4.24;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

    function get() public view returns (uint) {
        return storedData;
    }
}
【图片 remix compile.png】
```

编译合约

编写好合约后，可以使用remix的compile功能编译合约，检查合约中可能存在的语法错误。
在Seele wiki "Using the contract simulator"
小节中介绍了如何使用remix获得字节码。

在remix界面中，点击右侧的Run，Environment选择JavaScript VM，可以看到有默认的账户，gas limit，
下方即为我们编辑的合约名称SimpleStorage。点击Deploy可以测试合约的部署并得到字节码。\
【图片 run.png】

在remix中间panel的底部有日志窗口，显示部署合约的结果。展开对应的日志内容，可以看到status，transaction hash等信息。其中的input内容即为该合约对应的字节码。\
【图片 deployed.png】\

上述示例合约对应的字节码为：
0x6080604052348015610010576000080fd5b5060df8061001f6000396000f30060806040526004361060495760003 57c0100000000000000000000000000000000000000000000000000000000900463ffffffff16806360fe47b114604e5780636d4ce63c146078575b600080fd5b34801560595760008 0fd5b506076600480360381019080803590602001909291905050506a0565b005b34801560835760008 0fd5b50608a60aa565b6040518082815260200191505060405180910390f35b80600081905550505 65b600080549050905600a165627a7a723058208df15a24cad7025968f08e277ea08a607527ac4c4dc9d8ef22c5f1149ea8160d0029

部署合约

有了字节码之后，可以使用seele client工具部署合约。
你可以在seele release中找到相应平台下的软件包。
注意，部署合约与发送交易一样，你需要运行相应的Seele node程序。

本地测试环境

为保障提交到Seele主网的合约是可用的，建议先在本地运行Seele私链节点，
并基于私链进行合约部署和调用测试。

主网环境

本地测试环境通过后，你可以按照私链部署合约的步骤，将合约部署到Seele主网上。
唯一的不同是，你需要运行一个主网的Seele node程序，并保持和Seele主网的数据同步。

通过client工具部署合约

- 准备一个Seele账户，并确保有足够的余额部署合约。\
  这里我们使用分片1的账户0x987f6215c30f1505e0d42769c5472b410d6bf961，并且保存了账户keystore文件
- 使用sendtx命令部署合约

```
./client sendtx --amount 0 --from 0x987f6215c30f1505e0d42769c5472b410d6bf961.keystore --payload 0x6080604052348015610010576000080fd5b5060ec8061001f6000396000f3fe6080604052600436106049576000357c0100000000000000000000000000000000000000000000000000000000900463ffffffff16806360fe47b114604e5780636d4ce63c146085575b600080fd5b348015605957600080fd5b506083600480360360208110156 06e57600080fd5b8101908080359060200190929190505050 60ad565b005b348015609057600080fd5b50609760b7565b6040518082815260200191505060405180910390f35b8060008190555050565b600080549050905 6fea165627a7a7230582045fa1cc38960a931a2b1cf070ffa20bf25683c97838b5f32de5f37a7d6f5568d0029 --gas 100000
```

sendtx参数amount 为交易金额（单位Fan），对于合约部署通常可以设置为0；from为发送交易账户对应的keystore文件；
payload为合约字节码；gas为该笔交易所需最小gas值（默认为20000）。上述命令执行结果如下：

```
account: 0x987f6215c30f1505e0d42769c5472b410d6bf961, transaction nonce: 0
transaction sent successfully
{
    "Hash":
"0x80cb46ef3e3350e5c681397444f2b180624a9c1d534927a12e78cd689ef022fe",
    "Data": {
        "Type": 0,
        "From": "0x987f6215c30f1505e0d42769c5472b410d6bf961",
        "To": "0x0000000000000000000000000000000000000000",
        "Amount": 0,
        "AccountNonce": 0,
        "GasPrice": 10,
        "GasLimit": 200000,
        "Timestamp": 0,
        "Payload":
"0x6080604052348015610010576000080fd5b5060ec8061001f6000396000f3fe608060405
260043610604957600035c0100000000000000000000000000000000000000000000000000
0000000900463ffffffff16806360fe47b114604e5780636d4ce63c146085575b600080fd5b
348015605957600080fd5b506083600480360360208110156056e57600080fd5b8101908080
359060200190929190505050606ad565b005b34801560905760080fd550609760b7565b604
051808281526020019150506040518091039035b80600081905555050565b6000805490509
056fea165627a7a7230582045fa1cc38960a931a2b1cf070ffa20bf25683c97838b5f32de5
f37a7d6f5568d0029"
    },
    "Signature": {
        "Sig":
"KHi0wZMIeH4SIDtD0hLvDJp++I3H6C7HpFz5H9AEDbIxUDFydVbif9jQtXd9zeDlNfOdCG2x2
QwijGVqlFjItQE="
    }
}

```

- 根据交易Hash查看合约部署结果：

```
client getreceipt --hash
0x80cb46ef3e3350e5c681397444f2b180624a9c1d534927a12e78cd689ef022fe
```

得到如下结果：

```
{
    "contract": "0x47a990592190055cf8277d5d7dff933446edb0012",
    "failed": false,
    "poststate":
"0x7a6b4d2f7b09e1bdaacb3185c2013f361ea9dc0b05a233ecd268188098878874",
    "result":
"0x6080604052600436106049576000357c0100000000000000000000000000000000000000
0000000000000000000000900463ffffffff16806360fe47b114604e5780636d4ce63c1460855
75b600080fd5b348015605957600080fd5b506083600480360360208110156056e57600080f
d5b810190808035906020019092919050505060ad565b005b348015609057600080fd5b506
09760b7565b604051808281526020019150506040518091039035b8060008190555050565
b6000805490509056fea165627a7a7230582045fa1cc38960a931a2b1cf070ffa20bf25683
c97838b5f32de5f37a7d6f5568d0029",
```

```
    "totalFee": 837670,
    "txhash":
"0x80cb46ef3e3350e5c681397444f2b180624a9c1d534927a12e78cd689ef022fe",
    "usedGas": 83767
}
```

结果中failed为false表明合约部署成功，contract的值为合约地址

调用合约

使用sendTx调用合约

- 通过remix获得方法调用字节码\
  使用client sendtx命令调用合约需要提供合约调用方法的payload信息，可以通过remix获得。
  在remix右下方已部署的合约处，点击左侧小三角可以看到该合约包含的变量和方法。本示例合约包含set和get方法。
  在set右侧填入参数值，点击set，即可实现函数调用。日志窗口可以看到调用的结果。点击详情可查看调用详情，
  其中input的值即为方法调用的字节码。本示例调用set，设置变量值为21，如下图所示：
  【图片 callSet.png】

- 使用sendtx调用合约\
  如下的命令调用上述合约中的set方法，设置变量值为21。其中payload 为从remix获得的字节码，
  from为发起交易的账户，to为合约地址

```
client sendtx --amount 0   --payload
0x60fe47b1000000000000000000000000000000000000000000000000000000000000000015
--from 0x987f6215c30f1505e0d42769c5472b410d6bf961.keystore --to
0x47a99059219055cf8277d5d7dff933446edb0012
```

执行结果如下：

```
account: 0x987f6215c30f1505e0d42769c5472b410d6bf961, transaction nonce: 1
transaction sent successfully
{
    "Hash":
"0x44925f095e067c00660769d2488ab44a6eb9cf3183767d343f3005e7d60e98d6",
    "Data": {
        "Type": 0,
        "From": "0x987f6215c30f1505e0d42769c5472b410d6bf961",
        "To": "0x47a99059219055cf8277d5d7dff933446edb0012",
        "Amount": 0,
        "AccountNonce": 1,
        "GasPrice": 10,
        "GasLimit": 200000,
        "Timestamp": 0,
        "Payload":
"0x60fe47b1000000000000000000000000000000000000000000000000000000000000000001
```

```
5"
    },
    "Signature": {
        "Sig":
"m8tt0j10uDlhd06HO9M2iW0xQe5ZcX+i1N6D7IkbA91BG4/MMhsonTXUEleYyeKa+V02S96kz
d2QhgyMkdBDZQA="
    }
}
```

通过查询receipt信息，可以获取合约调用结果：

```
client getreceipt --hash
0x44925f095e067c00660769d2488ab44a6eb9cf3183767d343f3005e7d60e98d6
```

得到如下结果：

```
{
    "contract": "0x",
    "failed": false,
    "poststate":
"0x3f5f6e4d2747b50a55b9f86952d7eb6cd42976db80a624a3f2472af8508029d0",
    "result": "0x",
    "totalFee": 416950,
    "txhash":
"0x44925f095e067c00660769d2488ab44a6eb9cf3183767d343f3005e7d60e98d6",
    "usedGas": 41695
}
```

__【注意：调用合约的账户必须是与合约地址同一分片的，Seele目前不支持跨片合约调用】__

使用call调用合约方法

对于仅获取变量值而不改变合约状态的方法，可以通过call来调用。
获取对应方法payload信息和上述sendtx调用合约的方法类似。\
如调用示例合约中的get方法：

```
client call --payload 0x6d4ce63c --to
0x47a99059219055cf8277d5d7dff933446edb0012
```

可以得到如下结果：

```
{
    "contract": "0x",
    "failed": false,
    "poststate":
"0x44f2eb71f9044e8ff3620fae7adf1581a0d1f96cbefbaa546b824de8079c5969",
    "result":
"0x0000000000000000000000000000000000000000000000000000000000000015",
    "totalFee": 21696,
```

```
    "txhash":
"0xbaf65d3280c940d78040535ac925905fdeaefeaf4d210dcc5f0a9d5cd76767d3",
    "usedGas": 21696
}
```

可以看到，result显示的变量值为之前通过set方法设置的21。