享云链智能合约接入指南

智能合约部署与调用

编写合约代码

下面以一个简单的合约为例

```
pragma solidity ^0.5.5;

contract Test {
    uint public t;
    string public s;
    address public owner;

    constructor() public {
        owner = msg.sender;
    }

    modifier onlyOwner {
        require(msg.sender == owner, "only owner operate");
        _;
    }

    function setT(uint _t) public {
        t= _t;
    }

    function setS(string memory _s) public onlyOwner {
        s = _s;
    }

}
```

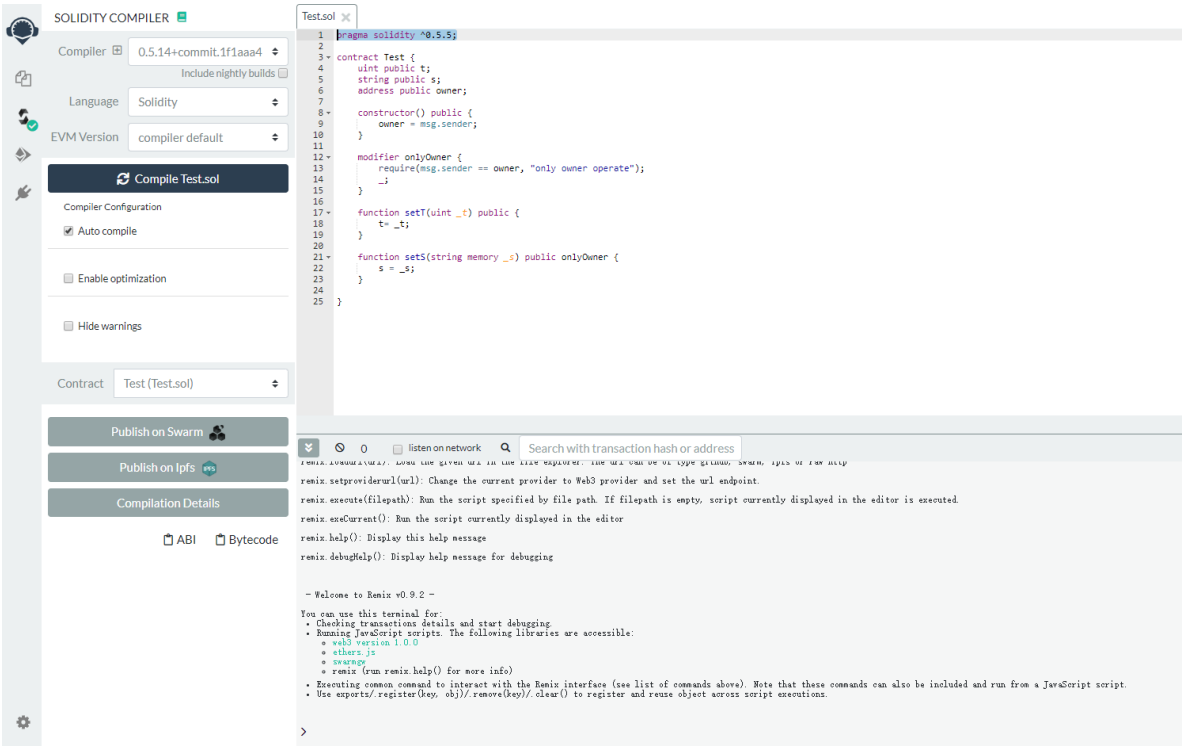
编译合约

编译合约的过程主要是为了获取部署合约所需的bytecode和合约调用所需的通用ABI。
编译的方式有很多，既可以选择使用在线编译工具如 remix 编译，也可以选择使用truffle、waffle等框架创建本地合约工程并编译。

下面简单介绍两种编译方式的使用过程：

使用remix编译合约

需要在 remix 右侧的plugin manager里添加 Solidity Compiler Plugin。
然后根据合约的需要选择合适的编译选项，比如 solc 的版本，EVM的版本以及优化选项等。
修改代码后点击 Compile 或者勾选自动编译选项，可以点击下面的Compilation Details按钮查看编译详情，或者直接点击下面的Bytecode和ABI按钮复制对应的内容。



使用truffle编译合约

1. 新建合约工程

```
$ npm i -g truffle
$ mkdir vote-contract
$ cd vote-contract
$ truffle init
```

2. 添加合约文件 Vote.sol
3. 修改编译选项 truffle-config.js

编译选项的修改主要是

```
compilers: {
    solc: {
        version: "0.5.8",    // Fetch exact version from solc-bin (default:
truffle's version)
        // docker: true,        // Use "0.5.1" you've installed locally with
docker (default: false)
        settings: {          // See the solidity docs for advice about
optimization and evmVersion
            optimizer: {
                enabled: true,
                runs: 200
            },
        //  evmVersion: "byzantium"
        }
    }
}
```

4. 执行编译 truffle compile
5. 从 build/Vote.json 下复制所需的bytecode和abi

部署合约与调用

由上一步得到的bytecode和ABI如下(编译环境不同得到的值可能不同)

```
// bytecode
0x608060405234801561001057600080fd5b50600280546001600160a01b0319163317905560016
0081905560408051808201909152600b8082527f68656c6c6f20776f726c64000000000000000000
000000000000000000000000602090920191825261006d9291906100735 65b5061010e565b828054
6001816001161561010002031660029004906000526020600020906 01f016020900481019282601f
106100b457805160ff1916838001178555610 0e1565b82800160010185558215610 0e1579182015b
828111156100e1578251825591602001919060010190610 0c6565b506100ed9291506100f1565b50
90565b61010b91905b808211156100ed5760008155600101610 0f7565b90565b6103be8061011d60
00396000f3fe608060405234801561001057600080fd5b50600436106100575760003560e01c8063
717020761461005c57806386b714e21461010457806 38da5cb5b14610181578063 92d0d153146101
a5578063f5f31941146101bf575b600080fd5b61010260048036036020811015610072576000 80fd
5b810190602081018135640100000000811115610 08d57600080fd5b82018360208201111561009f
57600080fd5b8035906020019184600183028401116401000000008311171561 00c157600080fd5b
91908080601f016020809104026020016040519081016040528093929190818152602001838380 82
843760009201919091525092955061 01dc945050505050565b005b61010c610247565b6040805160
20808252835181830152835191928392908301918501908083836000 5b83811015610146578181 01
5183820152602001610 12e565b50505050905090810190601f1680156101735780820380516001 83
6020036101000a031916815260200191505b509250505060405180910390f35b6101896102d4565b
604080516001600160a01b039092168252519081900360200190f35b6101ad6102e3565b60408051
918252519081900360200190f35b6101026004803603602081101561 01d557600080fd5b50356102
e9565b600254600160 0160a01b03163314610230576040805162461bcd60e51b815260206004820 1
526012602482015271 6f6e6c79206f776e6572206f706572617465 60701b604482015290519081 90
0360640190fd5b805161024390600190602084019061 02ee565b5050565b6001805460408051602 0
600284861615610100026000190190941693909304601f8101849004840282018401909252818152
929183018282801561 02cc5780601f106102a1576101008083540402835291602001916102cc565b
820191906000526020600020905b8154815290600101906020018083116102af578290036 01f1682
01915b505050505081565b600254600160 0160a01b031681565b60005481565b600055565b828054
6001816001161561010002031660029004906000526020600020906 01f016020900481019282601f
1061032f57805160ff1916838001178555610 35c565b8280016001018555821561035c579182015b
82811115610 35c578251825591602001919060010190610341565b5061036892915061036c565b50
90565b61038691905b8082111561036857600081556001016103725 65b9056fea265627a7a723158
20973e632ca02dad5527eb949827658d751ce7b3909648f007c641497ff12fd9e164736f6c634300
050c0032
```



```
//ABI
[{"constant": false,"inputs": [{"internalType": "string","name": "_s","type":
"string"}],"name": "setS","outputs": [],"payable": false,"stateMutability":
"nonpayable","type": "function"},{"constant": false,"inputs": [{"internalType":
"uint256","name": "_t","type": "uint256"}],"name": "setT","outputs":
[],"payable": false,"stateMutability": "nonpayable","type": "function"},
{"inputs": [],"payable": false,"stateMutability": "nonpayable","type":
"constructor"},{"constant": true,"inputs": [],"name": "owner","outputs":
[{"internalType": "address","name": "","type": "address"}],"payable":
false,"stateMutability": "view","type": "function"},{"constant": true,"inputs":
[],"name": "s","outputs": [{"internalType": "string","name": "","type":
"string"}],"payable": false,"stateMutability": "view","type": "function"},
{"constant": true,"inputs": [],"name": "t","outputs": [{"internalType":
"uint256","name": "","type": "uint256"}],"payable": false,"stateMutability":
"view","type": "function"}]
```


部署合约时需要一个有余额的账户来签署并发送交易。这里就需要用到前面启动的peer节点，以及节点内的账户来执行下面的操作。
从与peer节点的交互的方式上区分，可以从以下两种方式来执行部署合约和调用合约的流程：

1. 通过peer启动的rpc端口attach到geth控制台执行
2. 通过启动的wallet服务，发送请求到对应的接口来执行

通过peer节点的Geth控制台部署

这里的节点以本地测试网络节点为例。docker环境和启动过程参考快速入门。

部署合约

```
[root@b97bfae188b6]# /src/pack/lkchain/bin/lkchain attach http://127.0.0.1:11000

> TestContract = eth.contract(abi)
> personal.unlockAccount(eth.accounts()[0]) // 解锁要使用的发送账户
> ctx = TestContract.new({from: eth.accounts()[0], data: bytecode})
transactionHash:
"0x422747be10e702cd7338d13fa2d6a4a11010c203db88489e5861766778e40c74"
// 通过返回的交易hash查询部署的合约地址
>
eth.getTransactionReceipt('0x422747be10e702cd7338d13fa2d6a4a11010c203db88489e586
1766778e40c74')
{
  blockHash:
"0x510961350615d610728749d67ee50db4e72f9d7a684446a9488295ddcf35db3b",
  blockNumber: 151703,
  contractAddress: "0x132b86bbb129287ef3dce5193e88c66f7365f3c1",
  cumulativeGasUsed: 0,
  from: "0xa73810e519e1075010678d706533486d8ecc8000",
  gasUsed: 0,
  logs: [],
  logsBloom:
"0x00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000",
  status: "0x1",
  to: null,
  tokenAddress: "0x0000000000000000000000000000000000000000",
  transactionHash:
"0x422747be10e702cd7338d13fa2d6a4a11010c203db88489e5861766778e40c74",
  transactionIndex: 0
}
// 可以得到 contractAddress: "0x132b86bbb129287ef3dce5193e88c66f7365f3c1"
```

查询合约

```
> ctx = eth.contract(abi).at('0x132b86bbb129287ef3dce5193e88c66f7365f3c1')
> ctx.owner()
"0xa73810e519e1075010678d706533486d8ecc8000"
> ctx.t()
1
> ctx.s()
"hello world"
```

调用合约

```
> ctx.setT(100, {from: eth.accounts()[0]})
"0xd0dab5f5148835c936b6dc771f3ddc7d67d460141726f1a89f393d2134713490"
> ctx.t()
100
```

通过wallet接口部署与调用

本地启动的peer节点和wallet服务，可以使用wallet提供的接口实现合约部署和调用。

1. 解锁账户

```
$ curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","method":"personal_unlockAccount","params":
["0xa73810e519e1075010678d706533486d8ecc8000","1234",3600],"id":1}' -H 'Content-
Type:application/json'
{"jsonrpc":"2.0","id":1,"result":true}
```

2. getNonce

```
$ curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","id":"0","method":"ltk_getTransactionCount","params":
["0xa73810e519e1075010678d706533486d8ecc8000","latest"]}' -H 'Content-Type:
application/json'
{"jsonrpc":"2.0","id":"0","result":"0x0"}
```

3. 预估gas

```
curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","id":"0","method":"ltk_estimateGas","params":
[{"from":"0xa73810e519e1075010678d706533486d8ecc8000","value":"0x0","data":bytec
ode}]}' -H 'Content-Type: application/json'
{"jsonrpc":"2.0","id":"0","result":"0x5dff3"}
```

4. signTransaction 参数 gas 使用上一步得到的值，一般在结果上加一些，这里加20000；gasPrice 使用固定值

```
$ curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","id":"0","method":"ltk_signTransaction","params":
[{"from":"0xa73810e519e1075010678d706533486d8ecc8000","value":"0x0","data":bytec
ode,"nonce":"0x0","gas":"0x62e13","gasPrice":"0x174876e800"}]}' -H 'Content-
Type: application/json'
```

{"jsonrpc":"2.0","id":"0","result":
{"raw":"0xf905308085174876e80083062e138080b904db6080604052348015610010576000 80fd
5b5060028054600160016a01b0319163317905560016000819055604080518082019091526 00b80
82527f68656c6c6f20776f726c6400000000000000000000000000000000000000000000000000 6020909201
91825261006d929190610073565b5061010e565b828054600181600116156101000203166002 9004
906000526020600020906601f016020900481019282601f106100b457805160ff1916838001178555
6100e1565b828001600101855582156100e1579182015b828111156100e15782518255916020 0191
90600101906100c6565b506100ed9291506100f1565b5090565b61010b91905b808211156100ed57
600081556001016100f7565b90565b6103be8061011d6000396000f3fe6080604052348015610010
57600080fd5b506004361061005757600035 60e01c80637170207614610057806386b714e21461
01045780638da5cb5b146101815780639 2d0d153146101a5578063f5f31941146101bf575b600080
fd5b610102600480360360208110156100725760008 0fd5b810190602081018135640100000000811
111561008d57600080fd5b820183602082011115610 09f57600080fd5b8035906020019184600183
028401116401000000008311171561 00c157600080fd5b91908080601f0160208091040260200160
405190810160405280939291908181526020 0183838082843760009201919091525092955061 01dc
945050505050565b005b61010c610247565b6040805160208082528351818301528351919283929083
019185019080838360005b838110156101465781810151838201526020016101 2e565b50505050
905090810190601f1680156101735780820380516001836020036101000a031916815260200191505
b509250505060405180910390f35b6101896102d4565b604080516001600160a01b039092168252
519081900360200190f35b6101ad6102e3565b604080519182525190819003602001 90f35b610102
600480360360208110156101d557600080fd5b50356102e9565b600254600160a01b03163314
610230576040805162461bcd60e51b8152602060048201526012602482015271 6f6e6c79206f776e
6572206f7065726174656070 1b6044820152905190819003606401 90fd5b8051610243906001906020
8401906102ee565b5050565b6001805460408051602060028486161561010002600019019094 16
93909304601f810184900484028201840190925281815292918301828280156102cc5780601f1061
02a157610100808354040283529160200191610 2cc565b820191906000526020600020905b815481
529060010190602001808311610 2af57829003601f168201915b505050505081565b600254600160
0160a01b031681565b60005481565b600055565b828054600181600116156101000203166002 9004
906000526020600020906601f016020900481019282601f1061032f57805160ff1916838001178555
61035c565b828001600101855582156103 5c579182015b8281111561035c57825182559160200191
906001019061034156 5b5061036892915061036c565b5090565b61038691905b8082111561036857
600081556001016103725 65b9056fea265627a7a72315820973e632ca02dad5527eb949827658d75
1ce7b3909648f007c641497ff12fd9e164736f6c634300050c003282e3e6a09e94ef86ef39c0a2f7
3a14fa9196b0bd3015fd894c7d331e6581218e04260422a02b0c1424a908148c6aad73997af7a3d0
5634ec770de998565c85bb89ac529f61","tx":
{"nonce":"0x0","gasPrice":"0x174876e800","gas":"0x62e13","to":null,"value":"0x0"
,"input":"0x6080604052348015610010576000 80fd5b5060028054600160016a01b0319163317
905560016000819055604080518082019091526 00b8082527f68656c6c6f20776f726c6400000000
0000000000000000000000000000000000000060209092019182526 1006d929190610073565b5061010e
565b828054600181600116156101000203166002900490600052602060002090601f016020900481
019282601f106100b457805160ff1916838001178555561 00e1565b828001600101855582156100e1
579182015b828111156100e1578251825591602001919060010190610 0c6565b506100ed92915061
00f1565b5090565b61010b91905b808211156100ed576000815560010161 00f7565b90565b6103be
8061011d6000396000f3fe6080604052348015610010057600080fd5b5060043610610057576000 35
60e01c80637170207614610057806386b714e2146101045780638da5cb5b146101815780639 2d0
d153146101a5578063f5f31941146101bf575b600080fd5b610102600480360360208110156100 72
57600080fd5b810190602081018135640100000000811111561008d57600080fd5b8201836020820 1
111561009f57600080fd5b803590602001918460018302840111640100000000831117156100c157
600080fd5b91908080601f0160208091040260200160405190810160405280939291908181526020
0183838082843760009201919091525092955061 01dc945050505050565b005b61010c610247565b
604080516020808252835181830152835191928392908301918501908083836000 5b838110156101
465781810151838201526020016101 2e565b5050505090509081019060 1f1680156101735780820
380516001836020036101000a031916815260200191505b509250505060405180910390f35b610189
6102d4565b604080516001600160a01b039092168252519081900360200190f35b6101ad6102e356
5b604080519182525190819003602001 90f35b610102600480360360208110156101d557600080fd
5b50356102e9565b600254600160a01b031633146102305760404805162461bcd60e51b815260
206004820152601260248201527 16f6e6c79206f776e6572206f7065726174656070 1b6044820152
905190819003606401 90fd5b8051610243906001906020840190 6102ee565b5050565b6001805460

```
40805160206002848616156101000260001901909416939093046 01f810184900484028201840190
92528181529291830182828015610 2cc5780601f106102a157610100808354040283529160200191
6102cc565b8201919060005260206000020905b8154815290600101906020018083116102af578290
03601f168201915b505050505081565b600254600160016 0a01b031681565b60005481565b600055
565b828054600181600116156101000203166002900490600052602060002090601f016020900481
019282601f1061032f57805160ff191683800117855561035c565b828001600101855582156103 5c
579182015b8281111561035c57825182559160200191906001019061034 1565b5061036892915061
036c565b5090565b61038691905b808211156103685760008155600101610372565b9056fea26562
7a7a72315820973e632ca02dad5527eb949827658d751ce7b3909648f007c641497ff12fd9e16473
6f6c634300050c0032","v":"0xe3e6","r":"0x9e94ef86ef39c0a2f73a14fa9196b0bd3015fd89
4c7d331e6581218e04260422","s":"0x2b0c1424a908148c6aad73997af7a3d05634ec770de9985
65c85bb89ac529f61","hash":"0xe8ddee932a37d73635affcbcb55aaa6a13cb7f2261491315f57
a312887058eaf"}}}
```

5. sendRawTransaction 发送部署交易 params 取上一步得到的raw

```
$ curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","id":"0","method":"ltk_sendRawTransaction","params":
["0xf905308085174876e80083062e138080b904db6080604052348015610010576000 80fd5b5060
02805460016001600160a01b0319163317905560016000819055604080518082019091526 00b8082527f
68656c6c6f20776f726c6400000000000000000000000000000000000000000000000060209092019182 52
61006d929190610073565b5061010e565b828054600181600116156101000203166002900490600 0
52602060002090601f016020900481019282601f106100b457805160ff191683800117855561 00e1
565b828001600101855582156100e1579182015b828111156100e157825182559160200191906001
01906100c6565b506100ed9291506100f1565b5090565b61010b91905b808211156100ed57600081
556001016100f7565b90565b6103be8061011d6000396000f3fe6080604052348015610010576000
80fd5b506004361061005757600035 60e01c80637170207614610 05c57806386b714e21461010457
80638da5cb5b1461018157806392d0d153146101a5578063f5f31941146101bf575b600080fd5b61
0102600480360360208110156100725760008 0fd5b8101906020810181356401000000081111561
008d57600080fd5b8201836020820111156100 9f57600080fd5b8035906020019184600183028401
1164010000000831117156100c157600080fd5b91908080601f0160208091040260200160405190
810160405280939291908181526020018383808284376000920191909152509295506101dc945050
505050565b005b61010c610247565b604080516020808252835181830152835191928392908301 91
85019080838360005b838110156101465781810151838201526020016101 2e565b50505050905090
810190601f168015610173578082038051600183602003610100 0a031916815260200191505b5092
505050604051809103 90f35b6101896102d4565b604080516001600160 a01b039092168252519081
900360200190f35b6101ad6102e3565b604080519182525190819003602001 90f35b610102600480
360360208110156101d557600080fd5b50356102e9565b600254600160016 0a01b03163314610230
576040805162461bcd60e51b8152602060048201526026024820152716f6e6c79206f776e657220
6f70657261746560701b6044820152905190819003606401 90fd5b805161024390600190602084 01
906102ee565b5050565b6001805460408051602060028486161561010002600019019094169390 93
04601f810184900484028201840190925281815292918301828280156102cc5780601f106102a157
6101008083540402835291602001916102cc565b820191906000526020600020905b815481529060
01019060200180831161 02af57829003601f168201915b505050505081565b600254600160016 0a0
1b031681565b60005481565b600055565b828054600181600116156101000203166002900490600
052602060002090601f016020900481019282601f1061032f57805160ff191683800117855561035c
565b828001600101855582156103 5c579182015b8281111561035c57825182559160200191906001
0190610341565b506103689291506103 6c565b5090565b61038691905b808211156103685760008 1
556001016103 72565b9056fea265627a7a72315820973e632ca02dad5527eb949827658d751ce7b3
909648f007c641497ff12fd9e164736f6c634300050c003282e3e6a09e94ef86ef39c0a2f73a14fa
9196b0bd3015fd894c7d331e6581218e04260422a02b0c1424a908148c6aad73997af7a3d05634ec
770de998565c85bb89ac529f61"]}' -H 'Content-Type: application/json'
{"jsonrpc":"2.0","id":"0","result":"0xe8ddee932a37d73635affcbcb55aaa6a13cb7f2261
491315f57a312887058eaf"}
```

6. 根据hash查询交易

```
$ curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","id":"0","method":"ltk_getTransactionReceipt","params":
["0xe8ddee932a37d73635affcbcb55aaa6a13cb7f2261491315f57a312887058eaf"]}' -H
'Content-Type: application/json'
{"jsonrpc":"2.0","id":"0","result":
{"blockHash":"0x4c81ca8cbc30f6440c2fd628db6b6479a086f74c131cc9842199e57ff8a5dc0a
","blockNumber":"0xb","contractAddress":"0x25faf109d3c0514e4e021dd6baabfa1e8473e
97b","cumulativeGasUsed":"0x5dff3","from":"0xa73810e519e1075010678d706533486d8ec
c8000","gasUsed":"0x5dff3","logs":
[],"logsBloom":"0x00000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000","status":"0x1","to":null,"to
kenAddress":"0x0000000000000000000000000000000000000000","transactionHash":"0xe8
ddee932a37d73635affcbcb55aaa6a13cb7f2261491315f57a312887058eaf","transactionInde
x":"0x0"}}
```

status=0x1 表示交易成功，contractAddress:0x25faf109d3c0514e4e021dd6baabfa1e8473e97b 即部署得到的合约地址

7. 执行合约

合约ABI方法里 constant 类型为false的函数通过这种方式来执行。

执行合约需要用合约的ABI对执行的方法和参数做编码，这里选择使用ethers.js，也可以选择使用其他编码工具;
go语言的可以参考go-ethers pack/unpack实现; java语言的可以参考web3j。

```
const ethers = require("ethers")
const abi = [{"constant": false,"inputs": [{"internalType": "string","name":
"_s","type": "string"}],"name": "setS","outputs": [],"payable":
false,"stateMutability": "nonpayable","type": "function"},{"constant":
false,"inputs": [{"internalType": "uint256","name": "_t","type":
"uint256"}],"name": "setT","outputs": [],"payable": false,"stateMutability":
"nonpayable","type": "function"},{"inputs": [],"payable":
false,"stateMutability": "nonpayable","type": "constructor"},{"constant":
true,"inputs": [],"name": "owner","outputs": [{"internalType": "address","name":
"","type": "address"}],"payable": false,"stateMutability": "view","type":
"function"},{"constant": true,"inputs": [],"name": "s","outputs":
[{"internalType": "string","name": "","type": "string"}],"payable":
false,"stateMutability": "view","type": "function"},{"constant": true,"inputs":
[],"name": "t","outputs": [{"internalType": "uint256","name": "","type":
"uint256"}],"payable": false,"stateMutability": "view","type": "function"}]
let abiCoder = new ethers.utils.Interface(abi)
let setTData = abiCoder.functions['setT'].encode([10])
// 0xf5f3194100000000000000000000000000000000000000000000000000000000000000000a
```

再次按照上面的执行交易的流程执行一遍

- 获取nonce
- 预估gas
- 解锁账户，sign交易，参数from,to(合约地址),data(setTData),gas(预估gas),nonce
- 发送raw
- 根据hash查询交易结果

```
$ curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","id":"0","method":"ltk_getTransactionCount","params":
["0xa73810e519e1075010678d706533486d8ecc8000","latest"]}' -H 'Content-Type:
application/json'
// {"jsonrpc":"2.0","id":"0","result":"0x1"}

$ curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","id":"0","method":"ltk_estimateGas","params":
[{"from":"0xa73810e519e1075010678d706533486d8ecc8000","to":"0x25faf109d3c0514e4e
021dd6baabfa1e8473e97b",
"data":"0xf5f3194100000000000000000000000000000000000000000000000000000000000
0a"}]}' -H 'Content-Type: application/json'
// {"jsonrpc":"2.0","id":"0","result":"0x685f"}

$ curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","id":"0","method":"ltk_signTransaction","params":
[{"from":"0xa73810e519e1075010678d706533486d8ecc8000","to":"0x25faf109d3c0514e4e
021dd6baabfa1e8473e97b",
"data":"0xf5f3194100000000000000000000000000000000000000000000000000000000000
0a","nonce":"0x5","gas":"0x685f","gasPrice":"0x174876e800"}]}' -H 'Content-Type:
application/json'
// {"jsonrpc":"2.0","id":"0","result":
{"raw":"0xf88a0585174876e80082685f9425faf109d3c0514e4e021dd6baabfa1e8473e97b80a4
f5f3194100000000000000000000000000000000000000000000000000000000000000a82e3e5a0
8f7e45fb92a128133502d075c815f6620d59d62d789e8420618f8eb695ef9edca031244b40d077be
259538c0648fac53057f4c6b6147cb88436f7c5d4ce907b002","tx":
{"nonce":"0x5","gasPrice":"0x174876e800","gas":"0x685f","to":"0x25faf109d3c0514e
4e021dd6baabfa1e8473e97b","value":"0x0","input":"0xf5f3194100000000000000000000
0000000000000000000000000000000000000000a","v":"0xe3e5","r":"0x8f7e45fb92a1281
33502d075c815f6620d59d62d789e8420618f8eb695ef9edc","s":"0x31244b40d077be259538c0
648fac53057f4c6b6147cb88436f7c5d4ce907b002","hash":"0xfa83fc700f922586c0a01c1635
3b86923c4e6ef1657e10e1c3fe1d7e13c703cd"}}}

$ curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","id":"0","method":"ltk_sendRawTransaction","params":
["0xf88a0585174876e80082685f9425faf109d3c0514e4e021dd6baabfa1e8473e97b80a4f5f319
4100000000000000000000000000000000000000000000000000000000000000a82e3e5a08f7e45
fb92a128133502d075c815f6620d59d62d789e8420618f8eb695ef9edca031244b40d077be259538
c0648fac53057f4c6b6147cb88436f7c5d4ce907b002"]}' -H 'Content-Type:
application/json'
//
{"jsonrpc":"2.0","id":"0","result":"0xfa83fc700f922586c0a01c16353b86923c4e6ef165
7e10e1c3fe1d7e13c703cd"}

$ curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","id":"0","method":"ltk_getTransactionReceipt","params":
["0xfa83fc700f922586c0a01c16353b86923c4e6ef1657e10e1c3fe1d7e13c703cd"]}' -H
'Content-Type: application/json'
```

```
// {"jsonrpc":"2.0","id":"0","result":
{"blockHash":"0x01b82619a79efa935e89e227a6feffa805ce2caef6b25f331ffd43d08924c756
","blockNumber":"0x154eaa","contractAddress":null,"cumulativeGasUsed":"0x685f","
from":"0xa73810e519e1075010678d706533486d8ecc8000","gasUsed":"0x685f","logs":
[],"logsBloom":"0x00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000","status":"0x1","to":"0x25faf
109d3c0514e4e021dd6baabfa1e8473e97b","tokenAddress":"0x0000000000000000000000000
000000000000000","transactionHash":"0x0d14c499347a7cf920d76b10519c58ac3f679c685e
3eda8d05e36d72af796dec","transactionIndex":"0x0"}}
```

查询合约函数

合约ABI方法里 constant 类型为true的函数通过直接调用来执行查询。
下面查询 t 的值:

1. 对执行方法做编码

用上一步中的abiCoder

```
let tData = abiCoder.functions['t'].encode([])
// 0x92d0d153
```

2. 调用ethcall接口查询

```
$ curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","id":"0","method":"ltk_call","params":
[{"to":"0x25faf109d3c0514e4e021dd6baabfa1e8473e97b","data":"0x92d0d153"},
"latest"]}' -H 'Content-Type: application/json'
//
{"jsonrpc":"2.0","id":1,"result":"0x00000000000000000000000000000000000000000000
0000000000000000000a"}
```

3. 对结果进行解码

```
let t =
abiCoder.functions['t'].decode("0x0000000000000000000000000000000000000000000000
00000000000000000a")
// [BigNumber {_hex: "0x0a"}]
t.toString()
// 10
```

链克口袋执行合约

链克口袋是享云链为普通用户提供的账户管理、资产管理以及快捷签发交易的客户端工具。

按照前面执行的合约方法构造交易结构:

```
{
    to: "0x25faf109d3c0514e4e021dd6baabfa1e8473e97b",
    data:
"0xf5f31941000000000000000000000000000000000000000000000000000000000000000a",
    gas: "0x685f",
    value: 0
}
// 扩展字段 extension
{
    desc: '', // 交易描述
    callback: '', // 客户端发送交易后回调应用方后台链接
    cbData: ''  // 客户端回调应用方后台时带入参数
}
```

直接生成二维码

二维码内容格式如下：
`base64(ptitlubancommon://contract?to=tx.to&data=tx.data&value=tx.value&gas=tx.gas&desc=encodeURL(extension.desc)&callback=encodeURL(extension.callback)&cbData=encodeURL(extension.cbData))`
内容中不需要的字段可以省略。此种方式只适用于交易内容小于160字符的情况，当交易内容过长时，会导致生成的二维码过于密集，扫码设备识别率低；这时建议使用*二维码活码方式*。

根据交易内容拼接字符串得:

```
// base64(ptitlubancommon://contract?
to=0x25faf109d3c0514e4e021dd6baabfa1e8473e97b&data=0xf5f319410000000000000000000
00000000000000000000000000000000000000000000a&gas=26719)
//
cHRpdGx1YmFuY29tbW9uOi8vY29udHJhY3Q/dG89MHhhMzZlNmQxMDljMWE5ZWVlNDllMGY2NmQzYWY2
OWRlM2I2YWFmZGNmJmRhdGE9MHhmNWYzMTk0MTAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAw
MDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMGEmZ2FzPTI2NzE5
```

二维码活码方式

如果交易内容过长，直接生成二维码会导致很难识别。这时可以通过活码方式生成二维码：
`https://www.lianxiangcloud.com/download/guider/#/guider?action=encodeURL(actionURL)`
通过链克口袋扫描二维码后，链克口袋会GET请求`actionURL`，actionURL由开发者提供可请求服务，要求对应的response结构如下：

```
{
    "iRet":0,
    "sMsg": "ok",
    "data": {
        "type": 1,
        "contract": base64(key=value&key1=value1)
    })
}
```

上述字段中contract里字符串拼接的就是tx交易的字段。按照上述格式拼接合约交易如下：

```
//
base64(to=0x25faf109d3c0514e4e021dd6baabfa1e8473e97b&data=0xf5f31941000000000000000000000000000000000000000000000000000000000000000a&gas=26719)
//
dG89MHhhMzZlNmQxMDljMWE5ZWVlNDllMGY2NmQzYWY2OWRlM2I2YWFmZGNmJmRhdGE9MHhmNWYzMTk0MTAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMGEmZ2FzPTI2NzE5

{
    "iRet":0,
    "sMsg": "ok",
    "data": {
        "type": 1,
        "contract":
"dG89MHhhMzZlNmQxMDljMWE5ZWVlNDllMGY2NmQzYWY2OWRlM2I2YWFmZGNmJmRhdGE9MHhmNWYzMTk0MTAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMGEmZ2FzPTI2NzE5"
    })
}
```

构造上述response的请求链接如`http://xxx/orderid/3126908c00fe42129bf358756ea1aea2`
拼接二维码内容
`https://www.lianxiangcloud.com/download/guider/#/guider?action=http%3A%2F%2Fxxx%2Forderid%2F3126908c00fe42129bf358756ea1aea2`
这里由于请求是一个测试链接，所以就不展示生成的二维码，由开发者自行生成自己环境下的二维码测试。

应用唤醒链克口袋

应用方拼接交易并直接通过链接的形式唤醒链克口袋，传入交易参数。
链接格式:
`ptitlubancommon://contract?to=tx.to&data=tx.data&value=tx.value&gas=tx.gas&desc=encodeURL(extension.desc)&callback=encodeURL(extension.callback)&cbData=encodeURL(extension.cbData)` 内容中不需要的字段可以省略。根据交易内容拼接链接如下:
`ptitlubancommon://contract?`

```
to=0x25faf109d3c0514e4e021dd6baabfa1e8473e97b&data=0xf5f319410000000000000000000000
00000000000000000000000000000000000000000a&gas=26719
```