井通科技

井通合约编写部署帮助文档

井通科技
JINGTUM TECH

目录

井通科技
JINGTUM TECH

井通合约使用有两种方式：

1. 通过合约使用网站：http://contract.jingtum.com。

2.使用井通api。

1 井通合约脚本规范

编写井通合约需遵循以下规范才可保证合约正确执行。

1.1 编写合约需使用lua语言

但是lua脚本不支持lua的io操作和lua模块的引用操作，以下类似操作在合约编写中是不合法的：

1) print(‘hello’)

2) require("module")

1.2 合约固定基本格式

1)合约以result={};语句开始。该table类型确保合约有正确的返回值类型。合约执行的结果通过该table返回，该table类型中包含两个值，布尔类型：result[‘state’]和字符串类型：result[‘res’]。result[‘state’]中存放合约执行状态，true表示合约执行成功，false表示合约执行失败。result[‘res’]中存放合约执行结果或错误信息。当result[‘state’]为false时，存放错误信息，当result[‘state’]为true时，存放合约执行的结果。

2)所有合约函数的参数都是一个table类型。合约脚本中必须包含一个Init(t)函数。该函数的作用是存储合约执行中需要的一些参数。具体参数由合约具体功能决定。

2 通过合约网站使用合约步骤

2.1 合约部署步骤

井通科技
JINGTUM TECH

部署合约

* 账号　该字段填写拥有足够多swt的账号。确保能够足额支付合约部署费用和向合约账号支付足额swt。

输入以j开头的井通地址

jHb9CJAWyB4jr91VRWn96DkukG4bwdtyTh

* 私钥　账号字段填写的账号对应的私钥。

输入以s开头的井通私钥，注意保密

snoPBjXtMeMyMHUVTgbuqAfg1SUTb

* 发送数量(SWT)　账号字段填写的账号向合约账号支付的swt数量，最少25个。

部署合约成功后，会生成一个合约账号，需要向该账号发送至少10SWT，以激活该账户。

100

* 合约脚本　lua编写的合约脚本。

输入用lua语言编写的脚本，用于执行。

```
result={}; function Init(t) result=scGetAccountInfo(t) return result end; function foo(t) a={}
result=scGetAccountInfo(t) return result end
```

* 脚本参数　需要传递给lua脚本中Init函数的函数参数

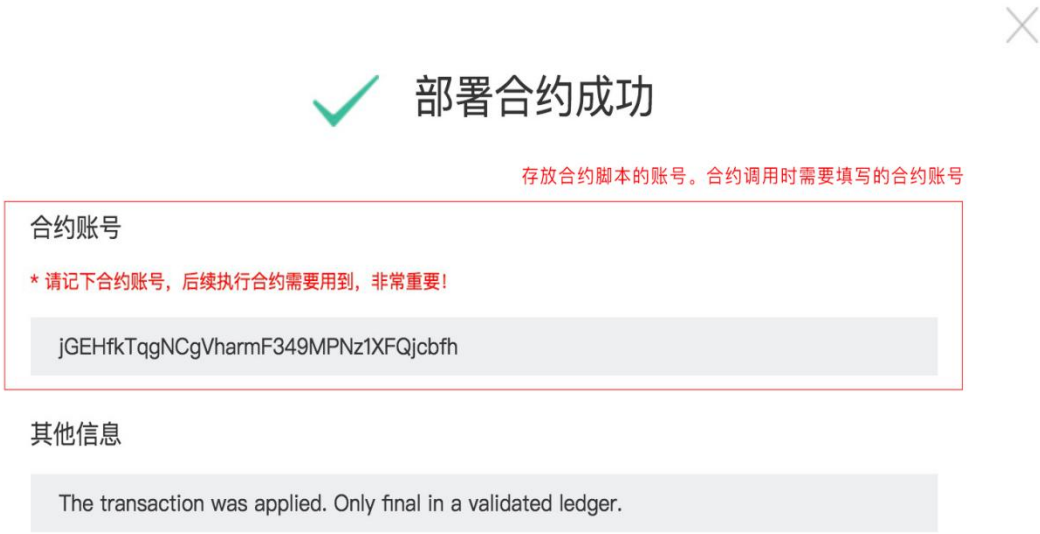
按脚本的执行顺序依次传入参数，每行只允许一个

jHb9CJAWyB4jr91VRWn96DkukG4bwdtyTh

+ 增加参数

部署合约

按上图的要求填写好字段后，点击部署合约按钮。如果部署成功会出现下面的界面：



如果部署失败上图会显示错误信息。部署成功后记录下合约账号，可以进行合约执行。

2.2 合约执行步骤

井通科技
JINGTUM TECH

执行合约

* 账号 该字段填写拥有足够多swt的账号，确保可以足额支付合约执行需要的费用。

输入以j开头的井通地址

jHb9CJAWyB4jr91VRWn96DkukG4bwdtyTh

* 私钥 账号字段填写的账号对应的私钥

输入以s开头的井通私钥，注意保密

snoPBjXtMeMyMHUVTgbuqAfg1SUTb

* 合约账号 合约部署成功返回的合约账号

输入部署合约成功后生成的合约账号

jb11xRYwzXgJZAh4PdGipgftGQLHHJD9u

* 合约脚本 会自动显示合约账号对应的合约脚本

输入合约账号后，自动从后台查询该合约的脚本，同时显示在这里，不可编辑

```
result={}; function Init(t) result=scGetAccountInfo(t) return result end; function foo(t) a={}
result=scGetAccountInfo(t) return result end
```

* 脚本参数 需要传递给合约函数名字段填写的函数的函数参数

按脚本的执行顺序依次传入参数，每行只允许一个

jHb9CJAWyB4jr91VRWn96DkukG4bwdtyTh

+ 增加参数

* 合约函数名 lua脚本中想要调用的合约函数名，Init函数除外。该实例中只能填foo

输入调用的合约函数名

foo

执行合约

按照上图填写好参数后，点击执行合约按钮。执行成功会出现如下界面

如果执行失败上图会显示失败原因。

3 通过井通API使用合约方式

3.1 部署合约

接口：/v2/accounts/**{:address}**/contract/deploy，POST方法
接口参数：

参数	类型	说明
address	String	井通钱包地址

提交参数详情：

参数	类型	说明
secret	String	用户的钱包私钥
amount	Integer	支付金额
payload	String	智能合约代码

可选接口参数：

参数	类型	说明
params	Array	合约参数

POST需要提交的参数格式如下：

```
{
  "secret": "snUaJxp2k4WFt5LCCtEx2zjThQhpT",
  "amount": 10,
  "params":  ["jsqRs9BDCjyTuRWEPZk3yHa4MFmRi9D834"],
  "payload": "result={};  function Init(t)  result=scGetAccountInfo(t)  return result  end;  funct
ion foo(t) a={} result=scGetAccountInfo(t)  return result  end"
}
```

例子：

http://localhost/v2/accounts/jsqRs9BDCjyTuRWEPZk3yHa4MFmRi9D834/contract/deploy

结果：

```
{
  "success": true,
  "status_code": "0",
  "ContractState": "contract:j43X58GWHGqWqQXM6b2BNr43AG8TjsmfYb",
  "engine_result": "tesSUCCESS",
  "engine_result_code": 0,
  "engine_result_message": "The transaction was applied. Only final in a validated ledger.",
  "tx_blob": "12001E220000000024000000001C2F215B24A1202400000000614000000000989680684000000000000000A
73210330E7FC9D56BB25D6893BA3F317AE5BCF33B3291BD63DB32654A313222F7FD0207447304502210 0A009376F5DCFA0
56F452C1B0703004A1D181D8479F908E353B2086867F4E89C20220100CCB09EDFEF416D1B611456BBEF4260E130E48E6AC
719EE8A3A89FE65AFE127FC066756E6374696F6E20496E6974282E2E2E2920613D7B7D20666F72206B2C7620696E206970
61697273287B2E2E2E7D2920646F20615B6B5D3D7620656E6420623D615B315D2072657475726E206163636F756E74696E
666F28622920656E643B202066756E6374696F6E20666F6F282E2E2E2920613D7B7D20666F72206B2C7620696E20697061
697273287B2E2E2E7D2920646F20615B6B5D3D7620656E6420623D615B315D2072657475726E206163636F756E74696E66
6F28622920656E648114B5F762798A53D543A014CAF8B297CFF8F2F937E8",
  "tx_json": {
    "Account": "jsqRs9BDCjyTuRWEPZk3yHa4MFmRi9D834",
   "Amount": "10000000",
   "Fee": "10",
   "Flags": 0,
   "Method": 0,
   "Payload": "726573756c743d7b7d3b202066756e6374696f6e20496e697428742920726573756c743d736347657
44163636f756e74496e666f28742920726573756c742020656e643b202066756e6374696f6e20666f6
f28742920613d7b7d20726573756c743d736347657441636 3f756e74496e666f28742920726573756e20726573756
c742020656e64",
   "Sequence": 28,
   "SigningPubKey": "0330E7FC9D56BB25D6893BA3F317AE5BCF33B3291BD63DB32654A313222F7FD020",
   "Timestamp": 559621281,
   "TransactionType": "ConfigContract",
   "TxnSignature": "3045022100A009376F5DCFA056F452C1B0703004A1D181D8479F908E353B2086867F4E89C20220
100CCB09EDFEF416D1B611456BBEF4260E130E48E6AC719EE8A3A89FE65AFE12",
   "hash": "4E00320F4D0FB5567ADB938A6463D62BA93ACF94C049972E2C4DCB01C6B57540",
  }
```

```
}
```

返回的结果信息：

参数	类型	说明
success	Boolean	请求结果
ContractState	String	生成的合约地址
engine_result	String	合约结果
engine_result_code	String	合约结果码
engine_result_message	String	合约结果说明
tx_blob	String	请求合约的blob
tx_json	Object	请求信息
Account	String	请求的井通地址
Amount	String	支付金额
Fee	String	交易费用，井通计价
Flags	String	交易标识
Method	String	交易方法：0表示部署，1表示调用
Payload	String	16进制智能合约代码
Sequence	String	单子号
SigningPubKey	String	签名公钥
Timestamp	Integer	交易时间，UNIXTIME
TransactionType	String	交易类型
TxnSignature	String	交易签名
hash	String	交易hash

3.2 调用合约

接口：/v2/accounts/**{:address}**/contract/call，POST方法

接口参数：

参数	类型	说明
address	String	井通钱包地址

提交参数详情：

参数	类型	说明
secret	String	用户的钱包私钥
destination	String	合约地址
foo	String	合约函数名

可选接口参数：

参数	类型	说明

params	Array	合约参数

POST需要提交的参数格式如下：


```
{
  "secret": "snUaJxp2k4WFt5LCCtEx2zjThQhpT",
  "destination": "jKotgzRHyoa7dywd7vf6LgFBXnv3K66zEg",
  "params":  ["jsqRs9BDCjyTuRWEPZk3yHa4MFmRi9D834"],
  "foo": "foo"
}
```


例子：

http://localhost/v2/accounts/jsqRs9BDCjyTuRWEPZk3yHa4MFmRi9D834/contract/call

结果：


```
{
  "success": true,
  "status_code": "0",
  "ContractState": {
    "Account": "jsqRs9BDCjyTuRWEPZk3yHa4MFmRi9D834",
    "Balance": "599999999829999910",
    "Flags": 0,
    "LedgerEntryType": "AccountRoot",
    "OwnerCount": 0,
    "PreviousTxnID": "4E00320F4D0FB5567ADB938A6463D62BA93ACF94C049972E2C4DCB01C6B57540",
    "PreviousTxnLgrSeq": 153616,
    "Sequence": 30,
    "index": "2B6AC232AA4C4BE41BF49D2459FA4A0347E1B543A4C92FCEE0821C0201E2E9A8",
 },
  "engine_result": "tesSUCCESS",
  "engine_result_code": 0,
  "engine_result_message": "The transaction was applied. Only final in a validated ledger.",
  "tx_blob": "12001E220000000024000000 01E2F215B2D3720240000000168400000000000000A73210330E7FC9D56BB
25D6893BA3F317AE5BCF33B3291BD63DB32654A313222F7FD020744730450221 00A2E6877B386732AB1857259705C38350
9812B60CAF8BF0615521BB06272625C6022052A3E44E6CAD493DED8A0ADF804BFE5B90BD68AD2B268CD42076FC410512FE
8B701103666F6F8114B5F762798A53D543A014CAF8B297CFF8F2F937E88314CE508BEB2DC80614229CB1E64F51A6537322
5835FAEB7012226A486239434A41577942346A7239315652576E3936446B756B4734627764747954 68E1F1",
  "tx_json": {
    "Account": "jsqRs9BDCjyTuRWEPZk3yHa4MFmRi9D834",
    "Args":  [
        {
            "Arg": {
              "Parameter": "6A486239434A41577942346A7239315652576E3936446B756B473462776474795468",
            }
        }
    ],
```

```
    "ContractMethod": "666F6F",
    "Destination": "jKotgzRHyoa7dywd7vf6LgFBXnv3K66zEg",
   "Fee": "10",
   "Flags": 0,
   "Method": 1,
   "Sequence": 30,
   "SigningPubKey": "0330E7FC9D56BB25D6893BA3F317AE5BCF33B3291BD63DB32654A313222F7FD020",
   "Timestamp": 559623479,
   "TransactionType": "ConfigContract",
   "TxnSignature": "3045022100A009376F5DCFA056F452C1B0703004A1D181D8479F908E353B2086867F4E89C20220
100CCB09EDFEF416D1B611456BBEF4260E130E48E6AC719EE8A3A89FE65AFE12",
   "hash": "4E00320F4D0FB5567ADB938A6463D62BA93ACF94C049972E2C4DCB01C6B57540",
  }
}
```


返回的结果信息：

参数	类型	说明
success	Boolean	请求结果
ContractState	Object	调用方信息
engine_result	String	合约结果
engine_result_code	String	合约结果码
engine_result_message	String	合约结果说明
tx_blob	String	调用合约的blob
tx_json	Object	调用信息
Account	String	井通地址
Args	Array	合约传入的参数
ContractMethod	String	合约方法
Destination	String	调用的合约地址
Fee	String	交易费用，井通计价
Flags	String	交易标识
Method	String	交易方法：0表示部署，1表示调用
Sequence	String	单子号
SigningPubKey	String	签名公钥
Timestamp	Integer	交易时间，UNIXTIME
TransactionType	String	交易类型
TxnSignature	String	交易签名
hash	String	交易hash