Development and deployment of Smart Contracts

Now, We use ' C + + ' to develop a smart contract for a mobile phone contact, □writing the user information to the chain before reading the information out.

Sample of How to Develop a Smart Contract

The code is as follows

> The functions of the following contract are □the storage and reading of `userName` and `userinfo`

- testRegUser.hpp

```
#define USER_NAME_LEN (20)
#define USER_INFO_LEN (20)


//@abi action reguser
struct UserInfo {
    char userName[USER_NAME_LEN];
    char userInfo[USER_INFO_LEN];
};


//@abi table userinfo:[index_type:string, key_names:userName, key_types:string]
struct UserBaseInfo {
    char userInfo[USER_INFO_LEN];
};
```

- testRegUser.cpp

```
#include "contractcomm.hpp"
#include "string.hpp"
#include "testRegUser.hpp"


#define PARAM_MAX_LEN (2048)

#define ERROR_PACK_FAIL (-1)
#define ERROR_UNPACK_FAIL (-2)
#define ERROR_SAVE_DB_FAIL (-3)
#define ERROR_METHOD_INVALID (-4)



static bool unpack_struct(MsgPackCtx *ctx, UserInfo *info)
{
```

```
    uint32_t size = 0;
    UNPACK_ARRAY(2)

    UNPACK_STR(info, userName, (USER_NAME_LEN+1))
    UNPACK_STR(info, userInfo, (USER_INFO_LEN+1))

    return 1;
}

static bool pack_struct(MsgPackCtx *ctx, UserBaseInfo *info)
{
    PACK_ARRAY16(1)
    PACK_STR16(info, userInfo )

    return 1;
}

void init()
{
}

int start(char* method)
{
    if (0 == strcmp("reguser", method))
    {
        char param[PARAM_MAX_LEN] = {0};
        uint32_t paramLen = 0;
        UserInfo userinfo = {{0}};

        paramLen = getParam(param, PARAM_MAX_LEN);

        MsgPackCtx ctx;
        msgpack_init(&ctx, (char*)param, paramLen);
        bool unpackSuc = unpack_struct(&ctx, &userinfo);
        if (!unpackSuc)
        {
            char pstr[] = "unpack userinfo failed";
            myprints(pstr);

            return ERROR_UNPACK_FAIL;
        }

        UserBaseInfo userBaseInfo = {{0}};

        strcpy(userBaseInfo.userInfo, userinfo.userInfo);

        msgpack_init(&ctx, (char*)param, paramLen);

        bool packSuc = pack_struct(&ctx, &userBaseInfo);
        if (!packSuc)
        {
            char pstr[] = "pack userbaseinfo failed";
            myprints(pstr);
```

```
            return ERROR_PACK_FAIL;
        }

        char objname[] = "userreginfo";

        uint32_t saveLen = setBinValue(objname, strlen(objname), userinfo.userName,
strlen(userinfo.userName), ctx.buf, ctx.pos);
        if (0 == saveLen)
        {
            char pstr[] = "save db failed";
            myprints(pstr);

            return ERROR_SAVE_DB_FAIL;
        }

        return 0;

    }
    else
    {
        char pstr[] = "invalid method";
        prints(pstr, strlen(pstr));
        return ERROR_METHOD_INVALID;
    }
}
```

Build a smart contract

Our smart contracts need to be compiled into a `. Wasm` format file for normal calls to be executed. So, the first step is to compile our smart contract code into a `. Wasm` file

- Download the compilation tool

```
git clone https://github.com/bottos-project/contract-tool-cpp.git
```

- Compile the contract

1. Enter the contract directory `Testreguser` and then run the following command to compile the contract

```
python ../gentool.py wasm testRegUser.cpp
```

At this point, `Testreguser.wast` and `testreguser.wasm` files are generated in the contract directory

◘2. Then run the following command to generate file `Testreguser.abi`

```
python ../gentool.py testRegUser.hpp
```

At this point, the contract directory generates a `Testreguser.abi` file

Deploy Smart Contract and ABI file

Once we have compiled the smart contract, we need the `bcli` tool to deploy the `. Wasm` file to the `Bottos` chain

BCLI Tool Usage

- Go to the `Bottos` project and deploy the compiled `wasm` file using the `bcli` tool

Copy the `BCLI` tool to the directory where the `wasm` and `Abi` files are located. Execute the following command

See how `bcli` is used

```
bcli --help
```

- Create a public private key pair

```
./bcli --servaddr 192.168.52.130:8689 wallet generatekey
```

The above command returns the public private key pair.

```
{
    "private_key": "773df18f6d7e1fa3fda1f2c36806bc40b20bbdd61ab59d00a2b47ecc4f9718e6",
    "public_key":
"04daec4f6b166ea21f5a3c4f8d50ef638ad0312cb01def711ba0ab350c10abeba8a137e5b5c4fffcbfdae4
d747595bb33f24fa2174abf8f631610d253977dfed23"
}
```

- Create an account by public key

```
./bcli --servaddr 192.168.52.130:8689 account create --username john --pubkey
04daec4f6b166ea21f5a3c4f8d50ef638ad0312cb01def711ba0ab350c10abeba8a137e5b5c4fffcbfdae4d
747595bb33f24fa2174abf8f631610d253977dfed23
```

If the following information is returned, the account creation is successful.

```
Create account: john Succeed
Trx:
{
    "version": 1,
    "cursor_num": 1071,
    "cursor_label": 1597958457,
    "lifetime": 1537171186,
    "sender": "bottos",
    "contract": "bottos",
    "method": "newaccount",
    "param": {
        "name": "john",
```

```
        "pubkey":
"04daec4f6b166ea21f5a3c4f8d50ef638ad0312cb01def711ba0ab350c10abeba8a137e5b5c4fffcbfdae4
d747595bb33f24fa2174abf8f631610d253977dfed23"
    },
    "param_bin":
"dc0002da00046a6f686eda00823034646165633466366231363665613231663561336334663864353065663633386164303331326362303164656637313162613061623335306331306162656261386131333765356235633466666663626664616534643734373539356262333366323466613231373461626638663633313631306432353339373737646665643233",
    "sig_alg": 1,
    "signature":
"fe62ac036c193a870ef1667f264b62c6c17c9d09f00958a9f6971931a187bbbf6b5c6129a162765cfcc02b
5da23d602bf4e19b84036bb88e58ce29ad38715117"
}
```

- See if your account creates results

```
./bcli --servaddr 192.168.52.130:8689 account get --username john
```

If you return a description, the creation succeeds.

```
Account: john
Balance: 0.00000000 BTO
```

- Deploy the `wasm` file to the chain

```
./bcli --servaddr 192.168.52.130:8689 contract deploy --name john --code
./testRegUser.wasm -abi testRegUser.abi
```

`Note`: The name of the contract here is the name of the account we created above

If the following information is returned, that is to say the contract was successfully deployed.

```
Deploy contract: john Succeed
Trx:
{
    "version": 1,
    "cursor_num": 45,
    "cursor_label": 3016045512,
    "lifetime": 1537497253,
    "sender": "john",
    "contract": "bottos",
    "method": "deploycode",
    "param": {
        "name": "john",
        "vm_type": 1,
        "vm_version": 1,
        "contract_code":
"0061736d010000000125066002 7f7f017f60027f7f0060067f7f7f7f7f7f017f60037f7f7f017f60000060
017f017f023c0403656e76086765745061726 16d000003656e76066d656d736574000303656e76067072696
e7473000103656e760b73657442..."
```

```
    },
    "param_bin":
"dc0004da00046a6f686ecc01cc01c50ea40061736d0100000001250660027f7f017f60027f7f0060067f7f
7f7f7f7f017f60037f7f7f017f60000060017f017f023c0403656e7608676574506172616d000003656e760
66d656d736574000303656e7606...",
    "sig_alg": 1,
    "signature":
"f2f96610b5ede1975359fd962b447264da4b1fa492869350a2a93540122dfd1d2b10f1bfe722ce484ea054
b59e2f8bd7120fa3ce17cc2e0de893997a2dcc18a5"
}
TrxHash: 7d3bea7da59d3a74b2d9625505bd64099efd2a509c20387ff28c924a12619250
signTrx: [183 153 239 97 104 48 205 123 133 153 174 121 88 251 238 86 212 200 22 143
253 84 33 161 96 37 163 152 184 164 190 69]
trx :%!(EXTRA *api.Transaction=version:1 cursor_num:45 cursor_label:3016045512
lifetime:1537497253 sender:"john" contract:"bottos" method:"deployabi"
param:"dc0002da00046a6f686ec503867b0a0922747970657322a205b5d2c0a0922737472756374732 23a
205b0a2020202020202020202020202020207b0a2020202020202020202020202020092 26e616d65223a20225
5736572496e666f222c0a2020202020202020202020202020092262617365223a2022222c0a202020202020
2020202020202020092266696656c6473223a207b0a2020202020202020202020202020090922757365724e6
16d65223a2022737472696e67222c0a2020202020202020202020202020090922757365724e666f223a20
22737472696e67220a2020202020202020202020202020202020202020207d0a2020202020202020202020202
020097d2c0a2020202020202020202020202020207b0a2020202020202020202020202020092 26e616d65223a
2022557365724261736 5496e666f222c0a2020202020202020202020202020092262617365223a2022222c0
a2020202020202020202020202020092266696656c6473223a207b0a2020202020202020202020202020909
2275736572496e666f223a2022737472696e67220a2020202020202020202020202020202020202020207d0a2
0202020202020202020202020097d0a202020202020205d2c0a0922616374696f6e73223a205b0a202020
2020202020202020207b0a2020202020202020202020202020092261637469 6f6e5f6e616d65223a202
272656775736572222c0a2020202020202020202020202020092274797065223a202255736572496e666f22
0a2020202020202020202020202020207d0a202020202020205d2c0a09227461626c6573223a205b0a2020202
0202020202020202020207b0a2020202020202020202020202020092274616 26c655f6e616d65223a202275
736572696e666f222c0a2020202020202020202020202020092269 6e6465785f74797065223a20227374726
96e67222c0a2020202020202020202020202020092266b65795f6e616d6573223a20205b0a20202020202020
2020202020202009092275736572 4e616d65220a2020202020202020202020202020092205d2c0a202020202
0202020202020092266b65795f7479706573223a20205b0a2020202020202020202020202020090922
737472696e67220a2020202020202020202020202020092205d2c0a2020202020202020202020202020092277
4797065223a2022557365724261736 5496e666f220a2020202020202020202020202020207d0a202020202020
205d0a7d0a" sig_alg:1 ){
    "errcode": 0,
    "msg": "trx receive succ",
    "result": {
        "trx": {
            "version": 1,
            "cursor_num": 45,
            "cursor_label": 3016045512,
            "lifetime": 1537497253,
            "sender": "john",
            "contract": "bottos",
            "method": "deployabi",
```

```
            "param":
"dc0002da00046a6f686ec503867b0a09227479706573223a205b5d2c0a092273747275637473223a205b0a
20202020202020202020202020207b0a20202020202020202020202020202009226e616d65223a20225573657
2496e666f222c0a20202020202020202020202020202009226261736 5223a2022222c0a2020202020202020
20202020200922666965 6c6473223a207b0a20202020202020202020202020200909227573657 24e616d652
23a2022737472696e67222c0a20202020202020202020202020200909227573657 24e666f223a20227374
72696e67220a20202020202020202020202020202020202020207d0a20202020202020202020202020200097
d2c0a20202020202020202020202020207b0a20202020202020202020202020202009226e616d65223a202255
736572426173656e666f222c0a20202020202020202020202020202009226261736 5223a2022222c0a20202
02020202020202020202020202009226669656c6473223a207b0a20202020202020202020202020200909227573
6572496e666f223a2022737472696e67220a20202020202020202020202020202020202020207d0a2020202
02020202020202020202020097d0a202020202020205d2c0a0922616374696f6e73223a205b0a202020202020
20202020202020207b0a20202020202020202020202020200922616374696f6e5f6e616d65223a202272656
775736572222c0a20202020202020202020202020202009227479706 5223a202255736572496e666f220a2020
20202020202020202020207d0a202020202020205d2c0a09227461626c6573223a205b0a2020202020202
02020202020202020207b0a20202020202020202020202020200922746 1626c655f6e616d65223a202275736572
696e666f222c0a20202020202020202020202020200922696e6465785f74797065223a2022737472696e672
22c0a20202020202020202020202020202009226b65795f6e616d6573223a20205b0a20202020202020202020
20202020090922757365724e616d65220a20202020202020202020202020202009205d2c0a202020202020202
02020202020202009226b65795f7479706573223a20205b0a20202020202020202020202020200909227374726
696e67220a20202020202020202020202020202009205d2c0a20202020202020202020202020202009227479706
5223a20225573657242617365496e666f220a20202020202020202020202020207d0a202020202020205d0a
7d0a",
            "sig_alg": 1,
            "signature":
"7d7ab2051ac3e2c419614849e036678e996a65999be556d5259087ff930c1c1a716d529c7062371e57a0fd
44894247b8e183b6cb1273dc517d8f5d8e64eaf8c5"
        },
        "trx_hash": "bfbeed13120730d77a9c3ea8aa0bb1f6bc8b4df78dc8fd306581ca8f59853617"
    }
}
```