**A链环境**：

http://192.168.7.69:10002/

"address" : "a00136bc18029f74548c1a1b9ae8a2449a43d93ce19184",

"private\_key" : "c0016efe1c273d0da3cac223836a191155a092b15b7105f2e4806be75608da6605bc21",

**通信合约地址**：

a00168babf35f0feac4854bb1fcc79d0235edfa87d0b60

**资产合约地址**：

a00230068c4eab8c26dd1cd140390fd09f9ffa9706845e

**B链环境：**

http://192.168.7.69:20002

"address" : "a001c3bbfce78bba8bfcb37113a84e95a3fd441a5622e3",

"private\_key" : "c0018d7939ec4085db3db6a2c698a6b30345301d81b6b53a700f4865a8c51f479149ee",

**通信合约地址**：

a0010cc417e4dfa7a952347980842d2d37f99a3ae190b0

**资产合约地址**：

a002121795274745cfa2d56577b15781c5fb5627458bc2

**公证人**：

a001ce7acd9c7cb6b9bd6acaeecaaa0d7ff240241a9d15

a001e5c2e5142c2b57e72dd5783b383eabfdf4f4e4f6bb

**from**

"address": "a0027e87e0c7ab5064fadcff92a6489d461aeb5432554d",

"private\_key": "c002565c91a4d53bbbb3ff8a2e38b55910746f2435430247fe3df780b81c043690856b",

**to**

"address": "a002b8467e03f6771c7e9dda09f9a7027b33fdf9a62386",

"private\_key": "c00203a3d0f7164b36abe290996f45a6ec88db4ffa0e92ef8162b819e91fd21f5d618e",

# 创建CPC合约

{

"items": [

{

"transaction\_json": {

"source\_address": "a001c3bbfce78bba8bfcb37113a84e95a3fd441a5622e3",

"nonce": 1,

"operations": [

{

"type": 1,

"create\_account": {

"dest\_address": "a0010cc417e4dfa7a952347980842d2d37f99a3ae190b0",

"priv": {

"master\_weight": 1,

"thresholds": {

"tx\_threshold": 1

}

},

"contract": {

"payload": "

'use strict';

const send\_proposal = 'send\_proposal\_';

const receive\_proposal = 'receive\_proposal\_';

const send\_relay = 'send\_relay';

const receive\_relay = 'receive\_relay';

const vote\_rate = 0.7;

const notary\_list = ['a001ce7acd9c7cb6b9bd6acaeecaaa0d7ff240241a9d15', 'a001e5c2e5142c2b57e72dd5783b383eabfdf4f4e4f6bb'];

const EXECUTE\_STATE\_INITIAL = 1;

const EXECUTE\_STATE\_PROCESSING = 2;

const EXECUTE\_STATE\_FAIL = 3;

const EXECUTE\_STATE\_SUCCESS = 4;

function findListIndex(array, key){

let i = 0;

for (i = 0; i < array.length; i += 1) {

if (array[i] === key) {

return i;

}

}

return false;

}

function doIssue(operation\_address,from,to,seq,amount){

let args = {

'action':'issue',

'from': from,

'to': to,

'cpcSeq':seq,

'amount': amount

};

let transaction = {

'operations': [{

'type': 3,

'payment':{

'dest\_address':operation\_address,

'input':JSON.stringify(args)

}

}]

};

let isSuccess = callBackDoOperation(transaction);

if (!isSuccess) {

//callBackLog('payment error: '+ JSON.stringify(transaction));

return false;

}

}

function doTransfer(operation\_address, from, to, amount) {

let args = {

'action': 'transfer',

'from': from,

'to': to,

'amount': amount

};

let transaction = {

'operations': [{

'type': 3,

'payment': {

'dest\_address': operation\_address,

'input': JSON.stringify(args)

}

}]

};

let isSuccess = callBackDoOperation(transaction);

if (!isSuccess) {

callBackLog('payment error: ' + JSON.stringify(transaction));

return false;

}

}

//initial relay

function initRelayContract(input){

let send\_relay\_data = {

'init\_seq': 0,

'complete\_seq': 0,

'notary\_list': notary\_list,

'f\_comm\_addr': thisAddress,

't\_comm\_addr': input.t\_comm\_addr,

'f\_chain\_id': 'CHAIN\_20190528\_A',

't\_chain\_id':'CHAIN\_20190528\_B'

};

let receive\_relay\_data = {

'init\_seq': 0,

'complete\_seq': 0,

'notary\_list': notary\_list,

'chain\_id': 'CHAIN\_20190528\_A'

};

let send\_relay\_db = callBackGetAccountMetaData(thisAddress, send\_relay);

let receive\_relay\_db = callBackGetAccountMetaData(thisAddress, receive\_relay);

if (send\_relay\_db === false&&receive\_relay\_db===false) {

let transaction = {

'operations': [{

'type': 4,

'set\_metadata': {

'key': send\_relay,

'value': JSON.stringify(send\_relay\_data)

}

},{

'type': 4,

'set\_metadata': {

'key': receive\_relay,

'value': JSON.stringify(receive\_relay\_data)

}

}]

};

let flag = callBackDoOperation(transaction);

if (!flag) {

throw 'Relay contract execution initialization failed';

}

}

}

//send cross chain tx

function checkSendOperationTx(input){

let metadata = callBackGetAccountMetaData(input.f\_assets\_addr,'tx\_'+input.seq);

let tx\_proof = {};

if (metadata) {

tx\_proof = JSON.parse(metadata.value);

}

let flag = tx\_proof.from === input.from && tx\_proof.to === thisAddress && tx\_proof.amount === input.amount&&tx\_proof.seq===input.seq;

if (flag === false) {

throw 'The operation contract does not cover this transaction';

}

}

function checkSendProposalTx(input){

let send\_proposal\_seq = callBackGetAccountMetaData(thisAddress,send\_proposal+input.f\_assets\_addr +'\_'+input.seq);

if(send\_proposal\_seq!==false){

throw 'Submit duplicate transaction';

}

}

function sendCrossChain(input){

checkSendProposalTx(input);

checkSendOperationTx(input);

let send\_relay\_db = callBackGetAccountMetaData(thisAddress, send\_relay);

let send\_relay\_data = JSON.parse(send\_relay\_db.value);

let send\_proposal\_seq = send\_relay\_data.init\_seq + 1;

let proposal = {

'seq': send\_proposal\_seq,

'amount': input.amount,

'from': input.from,

'to': input.to,

'f\_assets\_addr': input.f\_assets\_addr,

't\_assets\_addr': input.t\_assets\_addr,

'f\_comm\_addr': send\_relay\_data.f\_comm\_addr,

't\_comm\_addr': send\_relay\_data.t\_comm\_addr,

'f\_chain\_id': send\_relay\_data.f\_chain\_id,

't\_chain\_id': send\_relay\_data.t\_chain\_id

};

let proposal\_state = {

'proposal': proposal,

'state': EXECUTE\_STATE\_INITIAL,

'vote': [],

'vote\_count': 0

};

let relay\_data = {

'init\_seq': send\_proposal\_seq,

'complete\_seq': send\_relay\_data.complete\_seq,

'notary\_list': send\_relay\_data.notary\_list,

'f\_comm\_addr': send\_relay\_data.f\_comm\_addr,

't\_comm\_addr': send\_relay\_data.t\_comm\_addr,

'f\_chain\_id': send\_relay\_data.f\_chain\_id,

't\_chain\_id':send\_relay\_data.t\_chain\_id

};

let transaction = {

'operations': [{

'type': 4,

'set\_metadata': {

'key': send\_proposal + send\_proposal\_seq,

'value': JSON.stringify(proposal\_state)

}

},

{

'type': 4,

'set\_metadata': {

'key': send\_relay,

'value': JSON.stringify(relay\_data)

}

},

{

'type': 4,

'set\_metadata': {

'key': input.f\_assets\_addr +'\_'+input.seq,

'value':send\_proposal\_seq

}

}]

};

let isSuccess = callBackDoOperation(transaction);

if (!isSuccess) {

throw 'Contract execution cross chain transfer initialization failed';

}

}

//proccess send cross tx

function checkSendNotary(send\_relay\_db,send\_proposal\_db,input) {

let complete\_seq = send\_relay\_db.complete\_seq + 1;

let state = EXECUTE\_STATE\_INITIAL;

let vote\_array = [];

let i = 0;

for (i = 0; i < send\_proposal\_db.vote.length; i += 1) {

if (sender === send\_proposal\_db.vote[i][0]) {

throw 'A notary public may vote on each proposal only once';

}

}

for (i = 0; i < send\_proposal\_db.vote.length; i += 1) {

vote\_array.push([send\_proposal\_db.vote[i][0], send\_proposal\_db.vote[i][1]]);

if (input.state !== send\_proposal\_db.vote[i][1]) {

state = EXECUTE\_STATE\_FAIL;

break;

}

}

vote\_array.push([sender, input.state]);

if (state === EXECUTE\_STATE\_FAIL) {

let send\_proposal\_info = {

'proposal': send\_proposal\_db.proposal,

'state': EXECUTE\_STATE\_FAIL,

'vote': vote\_array,

'vote\_count': vote\_array.length

};

doTransfer(send\_proposal\_db.proposal.f\_assets\_addr, thisAddress, send\_proposal\_db.proposal.from, send\_proposal\_db.proposal.amount);

let send\_relay\_info = {

'init\_seq': send\_relay\_db.init\_seq,

'complete\_seq': complete\_seq,

'notary\_list': send\_relay\_db.notary\_list,

'f\_comm\_addr': send\_relay\_db.f\_comm\_addr,

't\_comm\_addr': send\_relay\_db.t\_comm\_addr,

'f\_chain\_id': send\_relay\_db.f\_chain\_id,

't\_chain\_id':send\_relay\_db.t\_chain\_id

};

let transaction = {

'operations': [{

'type': 4,

'set\_metadata': {

'key': send\_proposal + input.seq,

'value': JSON.stringify(send\_proposal\_info)

}

},

{

'type': 4,

'set\_metadata': {

'key': send\_relay,

'value': JSON.stringify(send\_relay\_info)

}

}]

};

let isSuccess = callBackDoOperation(transaction);

if (!isSuccess) {

throw 'Contract failure';

}

callBackLog(JSON.stringify(transaction));

return true;

}

}

function handleSendElect(send\_relay\_db,send\_proposal\_db,input){

let vote\_array = [];

let i = 0;

for (i = 0; i < send\_proposal\_db.vote.length; i += 1) {

vote\_array.push([send\_proposal\_db.vote[i][0], send\_proposal\_db.vote[i][1]]);

}

vote\_array.push([sender, input.state]);

let send\_proposal\_info = {

'proposal': send\_proposal\_db.proposal,

'state': EXECUTE\_STATE\_FAIL,

'vote': vote\_array,

'vote\_count': vote\_array.length

};

if (parseInt(send\_proposal\_info.vote\_count) < parseInt(send\_relay\_db.notary\_list.length \* vote\_rate + 0.5)) {

send\_proposal\_info.state = EXECUTE\_STATE\_PROCESSING;

let tx = {

'operations': [{

'type': 4,

'set\_metadata': {

'key': send\_proposal + input.seq,

'value': JSON.stringify(send\_proposal\_info)

}

}]

};

let isflag = callBackDoOperation(tx);

if (!isflag) {

throw 'Contract failure';

}

return;

}

let send\_relay\_info = {

'init\_seq': send\_relay\_db.init\_seq,

'complete\_seq': send\_relay\_db.complete\_seq+1,

'notary\_list': send\_relay\_db.notary\_list,

'f\_comm\_addr': send\_relay\_db.f\_comm\_addr,

't\_comm\_addr': send\_relay\_db.t\_comm\_addr,

'f\_chain\_id': send\_relay\_db.f\_chain\_id,

't\_chain\_id':send\_relay\_db.t\_chain\_id

};

switch (input.state) {

case EXECUTE\_STATE\_FAIL:

send\_proposal\_info.state = EXECUTE\_STATE\_FAIL;

doTransfer(send\_proposal\_db.proposal.f\_assets\_addr, thisAddress, send\_proposal\_db.proposal.from, send\_proposal\_db.proposal.amount);

break;

case EXECUTE\_STATE\_SUCCESS:

send\_proposal\_info.state = EXECUTE\_STATE\_SUCCESS;

break;

default:

throw 'Return result exception';

}

let transaction = {

'operations': [{

'type': 4,

'set\_metadata': {

'key': send\_proposal + input.seq,

'value': JSON.stringify(send\_proposal\_info)

}

},

{

'type': 4,

'set\_metadata': {

'key': send\_relay,

'value': JSON.stringify(send\_relay\_info)

}

}]

};

let isSuccess = callBackDoOperation(transaction);

if (!isSuccess) {

throw 'Contract failure';

}

callBackLog(JSON.stringify(send\_proposal\_info));

}

function onSendProposalEvent(input) {

let relay = callBackGetAccountMetaData(thisAddress, send\_relay);

if (relay === false) {

throw 'Channel information does not exist';

}

let send\_relay\_db = JSON.parse(relay.value);

let complete\_seq = send\_relay\_db.complete\_seq + 1;

if (findListIndex(send\_relay\_db.notary\_list, sender) === false) {

throw 'The operator is not a notary public';

}

if (input.seq !== complete\_seq) {

throw 'Inconsistent timing of communication information processing';

}

let proposal = callBackGetAccountMetaData(thisAddress, send\_proposal + input.seq);

if (proposal === false) {

throw 'The proposal does not exist';

}

let send\_proposal\_db = JSON.parse(proposal.value);

if (checkSendNotary(send\_relay\_db,send\_proposal\_db,input) === true) {

return;

}

handleSendElect(send\_relay\_db,send\_proposal\_db,input);

}

//handle receive cross tx

function initReceiveProposalEvent(receive\_relay\_db,proposal,input){

let proposal\_vote = [];

let origin\_proposal = [sender,proposal.seq,proposal.amount,proposal.from,proposal.to,proposal.f\_assets\_addr,proposal.t\_assets\_addr,proposal.f\_comm\_addr,proposal.t\_comm\_addr,proposal.f\_chain\_id,proposal.t\_chain\_id];

proposal\_vote.push(origin\_proposal);

let receive\_proposal\_info = {

'proposals': proposal\_vote,

'state': EXECUTE\_STATE\_INITIAL,

'vote\_count': proposal\_vote.length

};

let receive\_relay\_info = {

'init\_seq': receive\_relay\_db.init\_seq+1,

'complete\_seq': receive\_relay\_db.complete\_seq,

'notary\_list': receive\_relay\_db.notary\_list,

'chain\_id': receive\_relay\_db.chain\_id

};

let transaction = {

'operations': [{

'type': 4,

'set\_metadata': {

'key': receive\_proposal + input.seq,

'value': JSON.stringify(receive\_proposal\_info)

}

},

{

'type': 4,

'set\_metadata': {

'key': receive\_relay,

'value': JSON.stringify(receive\_relay\_info)

}

}]

};

let isSuccess = callBackDoOperation(transaction);

if (!isSuccess) {

throw 'Contract failure';

}

}

function checkRecceiveNotary(receive\_relay\_db,receive\_proposal\_db,proposal,input){

let proposal\_vote = [];

let state = EXECUTE\_STATE\_INITIAL;

proposal\_vote = receive\_proposal\_db.proposals;

let i = 0;

for (i = 0; i < receive\_proposal\_db.proposals.length; i += 1) {

if (sender === receive\_proposal\_db.proposals[i][0]) {

throw 'A notary public may vote on each proposal only once';

}

}

for (i = 0; i < receive\_proposal\_db.proposals.length; i += 1) {

let flag = input.seq!==receive\_proposal\_db.proposals[i][1]||input.amount!==receive\_proposal\_db.proposals[i][2]||

input.from !==receive\_proposal\_db.proposals[i][3]||input.to!==receive\_proposal\_db.proposals[i][4]||

input.f\_assets\_addr!==receive\_proposal\_db.proposals[i][5]||input.t\_assets\_addr!==receive\_proposal\_db.proposals[i][6]||

input.f\_comm\_addr!==receive\_proposal\_db.proposals[i][7]||input.t\_comm\_addr!==receive\_proposal\_db.proposals[i][8]||

input.f\_chain\_id!==receive\_proposal\_db.proposals[i][9]||input.t\_chain\_id!==receive\_proposal\_db.proposals[i][10];

if (flag===true) {

state = EXECUTE\_STATE\_FAIL;

break;

}

}

let origin\_proposal = [sender,proposal.seq,proposal.amount,proposal.from,proposal.to,proposal.f\_assets\_addr,proposal.t\_assets\_addr,proposal.f\_comm\_addr,proposal.t\_comm\_addr,proposal.f\_chain\_id,proposal.t\_chain\_id];

proposal\_vote.push(origin\_proposal);

let receive\_proposal\_info = {

'proposals': proposal\_vote,

'state': state,

'vote\_count': proposal\_vote.length

};

if(state === EXECUTE\_STATE\_INITIAL){

return false;

}

let receive\_relay\_info = {

'init\_seq': receive\_relay\_db.init\_seq,

'complete\_seq': receive\_relay\_db.complete\_seq+1,

'notary\_list': receive\_relay\_db.notary\_list,

'chain\_id': receive\_relay\_db.chain\_id

};

let transaction = {

'operations': [{

'type': 4,

'set\_metadata': {

'key': receive\_proposal + input.seq,

'value': JSON.stringify(receive\_proposal\_info)

}

},

{

'type': 4,

'set\_metadata': {

'key': receive\_relay,

'value': JSON.stringify(receive\_relay\_info)

}

}]

};

let isSuccess = callBackDoOperation(transaction);

if (!isSuccess) {

throw 'Contract failure';

}

}

function handleReceiveElect(receive\_relay\_db,receive\_proposal\_db,proposal,input){

let proposal\_vote = [];

let state = EXECUTE\_STATE\_INITIAL;

proposal\_vote = receive\_proposal\_db.proposals;

let origin\_proposal = [sender,proposal.seq,proposal.amount,proposal.from,proposal.to,proposal.f\_assets\_addr,proposal.t\_assets\_addr,proposal.f\_comm\_addr,proposal.t\_comm\_addr,proposal.f\_chain\_id,proposal.t\_chain\_id];

proposal\_vote.push(origin\_proposal);

let receive\_proposal\_info = {

'proposals':proposal\_vote,

'state': EXECUTE\_STATE\_PROCESSING,

'vote\_count': proposal\_vote.length

};

let receive\_relay\_info = {

'init\_seq': receive\_relay\_db.init\_seq,

'complete\_seq': receive\_relay\_db.complete\_seq+1,

'notary\_list': receive\_relay\_db.notary\_list,

'chain\_id': receive\_relay\_db.chain\_id

};

if (parseInt(receive\_proposal\_info.vote\_count) < parseInt(receive\_relay\_db.notary\_list.length \* vote\_rate + 0.5)) {

let tx = {

'operations': [{

'type': 4,

'set\_metadata': {

'key': receive\_proposal + input.seq,

'value': JSON.stringify(receive\_proposal\_info)

}

}]

};

let isflag = callBackDoOperation(tx);

if (!isflag) {

throw 'Contract failure';

}

return;

}

receive\_proposal\_info.state = EXECUTE\_STATE\_SUCCESS;

doIssue(input.t\_assets\_addr,thisAddress,input.to,input.seq,input.amount);

let transaction = {

'operations': [{

'type': 4,

'set\_metadata': {

'key': receive\_proposal + input.seq,

'value': JSON.stringify(receive\_proposal\_info)

}

},

{

'type': 4,

'set\_metadata': {

'key': receive\_relay,

'value': JSON.stringify(receive\_relay\_info)

}

}]

};

let isSuccess = callBackDoOperation(transaction);

if (!isSuccess) {

throw 'Contract failure';

}

}

function onReceiveProposalEvent(input){

let relay = callBackGetAccountMetaData(thisAddress, receive\_relay);

if (relay === false) {

throw 'Relay information does not exist';

}

let origin\_proposal = callBackGetAccountMetaData(thisAddress, receive\_proposal+input.seq);

let receive\_relay\_db = JSON.parse(relay.value);

let complete\_seq = receive\_relay\_db.complete\_seq + 1;

if (findListIndex(receive\_relay\_db.notary\_list, sender) === false) {

throw 'The operator is not a notary public';

}

if( input.t\_chain\_id!==receive\_relay\_db.chain\_id||input.t\_comm\_addr!==thisAddress){

throw 'args error';

}

if (input.seq !== complete\_seq) {

throw 'Inconsistent timing of communication information processing';

}

let proposal = {

'seq': input.seq,

'amount': input.amount,

'from': input.from,

'to': input.to,

'f\_assets\_addr': input.f\_assets\_addr,

't\_assets\_addr': input.t\_assets\_addr,

'f\_comm\_addr': input.f\_comm\_addr,

't\_comm\_addr': input.t\_comm\_addr,

'f\_chain\_id': input.f\_chain\_id,

't\_chain\_id': input.t\_chain\_id

};

if(origin\_proposal===false){

initReceiveProposalEvent(receive\_relay\_db,proposal,input);

return ;

}

let receive\_proposal\_db = JSON.parse(origin\_proposal.value);

if(checkRecceiveNotary(receive\_relay\_db,receive\_proposal\_db,proposal,input)!==false){

return;

}

handleReceiveElect(receive\_relay\_db,receive\_proposal\_db,proposal,input);

}

function main(inputData) {

callBackLog('inputData:' + inputData);

let input;

try {

input = JSON.parse(inputData);

} catch(error) {

return;

}

let action = input.

function;

switch (action) {

case 'initRelayContract':

initRelayContract(input);

break;

case 'sendCrossChain':

sendCrossChain(input);

break;

case 'onSendProposalEvent':

onSendProposalEvent(input);

break;

case 'onReceiveProposalEvent':

onReceiveProposalEvent(input);

break;

default:

throw 'Invalid operation type';

}

}

"

}

}

}

]

},

"private\_keys": [

"c0018d7939ec4085db3db6a2c698a6b30345301d81b6b53a700f4865a8c51f479149ee"

]

}

]

}

# 2.创建CAC合约

{

"items": [{

"transaction\_json": {

"source\_address": "a00136bc18029f74548c1a1b9ae8a2449a43d93ce19184",

"nonce": 168,

"operations": [{

"type": 1,

"create\_account": {

"dest\_address": "a00230068c4eab8c26dd1cd140390fd09f9ffa9706845e",

"priv": {

"master\_weight": 1,

"thresholds": {

"tx\_threshold": 1

}

},

"contract": {

"payload": "

'use strict';

function currentTimeMills() {

let consensus = consensusValue;

return parseInt(consensus.close\_time / 1000);

}

function getAmount(address) {

let metadata = callBackGetAccountMetaData(thisAddress, address);

if (metadata) {

return metadata.value;

}

return 0;

}

function getSeq() {

let metadata = callBackGetAccountMetaData(thisAddress, 'seq');

if (metadata) {

return parseInt(metadata.value);

}

return 0;

}

/\*\*

\* 设置账户metadata

\*/

function setMetaData(key, value)

{

//assert(typeof key === 'string', 'Args type error. key must be a string.');

if(value === undefined){

return false;

}

let transaction = {

'operations': [{

'type': 4,

'set\_metadata': {'key': key, 'value': JSON.stringify(value )}

}]

};

let isSuccess = callBackDoOperation(transaction);

if (!isSuccess) {

return false;

}

return true;

}

function issue(input) {

let data = input;

let amount = data.amount;

if (amount <= 0) {

throw '金额必须大于0';

}

let from = data.from;

let to = data.to;

let seq = getSeq() + 1;

//let seq = transactions.length;

let issue\_data = {

'from': from,

'to': to,

'amount': amount,

'seq': seq,

'actionTime': currentTimeMills()

};

if(data.hasOwnProperty('cpcSeq')){

issue\_data.cpcSeq = data.cpcSeq;

}

//transactions.push(issue\_par);

let transaction = {

'operations': [{

'type': 4,

'set\_metadata': {'key': to, 'value': amount}

}, {

'type': 4,

'set\_metadata': {'key': 'seq', 'value': seq}

}, {

'type': 4,

'set\_metadata': {'key': 'tx\_'+seq, 'value': JSON.stringify(issue\_data)}

}]

};

let isSuccess = callBackDoOperation(transaction);

if (!isSuccess) {

throw '合约执行登记失败';

}

}

function transfer(input) {

let data = input;

if(sender !== data.from){

throw '非本人进行的转账';

}

let transferAmount = data.amount;

if (transferAmount <= 0) {

throw '转出金额必须大于0';

}

let to = data.to;

let from = data.from;

let balance\_to = getAmount(to);

let balance\_from = getAmount(from);

if (balance\_from <= 0) {

throw '转让人余额不足';

}

if (balance\_from < transferAmount) {

throw '转让金额必须大于等于转让金额';

}

//let transactions = getTransactions();

let seq = getSeq() + 1;

let transfer\_data = {

'from': from,

'to': to,

'amount': transferAmount,

'seq': seq,

'actionTime': currentTimeMills()

};

let transaction = {

'operations': [{

'type': 4,

'set\_metadata': {'key': from, 'value': parseInt(balance\_from) - transferAmount}

}, {

'type': 4,

'set\_metadata': {'key': to, 'value': parseInt(balance\_to) + transferAmount}

}, {

'type': 4,

'set\_metadata': {'key': 'seq', 'value': seq}

},{

'type': 4,

'set\_metadata': {'key': 'tx\_'+seq, 'value': JSON.stringify(transfer\_data)}

}]

};

let isSuccess = callBackDoOperation(transaction);

if (!isSuccess) {

throw '合约执行转让失败';

}

}

function main(inputData) {

callBackLog('inputData: ' + inputData);

let input;

try {

input = JSON.parse(inputData);

} catch (error) {

return;

}

let action = input.action;

switch (action) {

case 'issue':

issue(input);

break;

case 'transfer':

transfer(input);

break;

default:

throw '无效的操作类型';

}

}

"

}

}

}]

},

"private\_keys": ["c0016efe1c273d0da3cac223836a191155a092b15b7105f2e4806be75608da6605bc21"]

}]

}

# 3.合约内发行资产

{

"items": [

{

"transaction\_json": {

"source\_address": "a00136bc18029f74548c1a1b9ae8a2449a43d93ce19184",

"nonce": 6,

"operations": [

{

"type": 3,

"payment": {

"dest\_address": "a00230068c4eab8c26dd1cd140390fd09f9ffa9706845e",

"input": "{\"action\":\"issue\",\"from\":\"a00136bc18029f74548c1a1b9ae8a2449a43d93ce19184\",\"to\":\"a0027e87e0c7ab5064fadcff92a6489d461aeb5432554d\",\"amount\":10000000}"

}

}

]

},

"private\_keys": [

"c0016efe1c273d0da3cac223836a191155a092b15b7105f2e4806be75608da6605bc21"

]

}

]

}

# 4.转移资产

{

"items": [

{

"transaction\_json": {

"source\_address": "a0027e87e0c7ab5064fadcff92a6489d461aeb5432554d",

"nonce": 1,

"operations": [

{

"type": 3,

"payment": {

"dest\_address": "a00230068c4eab8c26dd1cd140390fd09f9ffa9706845e",

"input": "{\"action\":\"transfer\",\"from\":\"a0027e87e0c7ab5064fadcff92a6489d461aeb5432554d\",\"to\":\"a00168babf35f0feac4854bb1fcc79d0235edfa87d0b60\",\"amount\":50}"

}

}

]

},

"private\_keys": [

"c002565c91a4d53bbbb3ff8a2e38b55910746f2435430247fe3df780b81c043690856b"

]

}

]

}

# 5.发起跨链交易

{

"items": [

{

"transaction\_json": {

"source\_address": "a0027e87e0c7ab5064fadcff92a6489d461aeb5432554d",

"nonce": 2,

"operations": [

{

"type": 3,

"payment": {

"dest\_address": "a00168babf35f0feac4854bb1fcc79d0235edfa87d0b60",

"input": "{\"function\":\"sendCrossChain\",\"f\_assets\_addr\":\"a00230068c4eab8c26dd1cd140390fd09f9ffa9706845e\",\"from\":\"a0027e87e0c7ab5064fadcff92a6489d461aeb5432554d\",\"to\":\"a002b8467e03f6771c7e9dda09f9a7027b33fdf9a62386\",\"amount\":50,\"t\_assets\_addr\":\"a002121795274745cfa2d56577b15781c5fb5627458bc2\",\"seq\":2}"

}

}

]

},

"private\_keys": [

"c002565c91a4d53bbbb3ff8a2e38b55910746f2435430247fe3df780b81c043690856b"

]

}

]

}

# 6.查看结果

用户操作查询

<http://192.168.7.69:10002/getAccount?address=a00230068c4eab8c26dd1cd140390fd09f9ffa9706845e>

<http://192.168.7.69:20002/getAccount?address=a002121795274745cfa2d56577b15781c5fb5627458bc2>

跨链操作查询

<http://192.168.7.69:10002/getAccount?address=a00168babf35f0feac4854bb1fcc79d0235edfa87d0b60>

<http://192.168.7.69:20002/getAccount?address=a0010cc417e4dfa7a952347980842d2d37f99a3ae190b0>