中山大学数据科学与计算机学院本科生实验报告

（2019 年秋季学期）

课程名称：区块链原理与技术　　　　　　任课教师：郑子彬

年级	2017 级	专业（方向）	软件工程
学号	17343072	姓名	廖婕
电话	15013027902	Email	1528504591@qq.com
开始日期	2019.12.6	完成日期	2019.12.13

一、项目背景

目前传统供应链金融存在诸多痛点：信用体系不完善，中小企业融资难；供应链条不互通，贸易信息不透明；操作方法局限多，流转保理确权难等等。本项目致力于基于已有的开源区块链系统 FISCO-BCOS，以联盟链为主，开发基于区块链或区块链智能合约的供应链金融平台，实现供应链应收账款资产的溯源、流转。

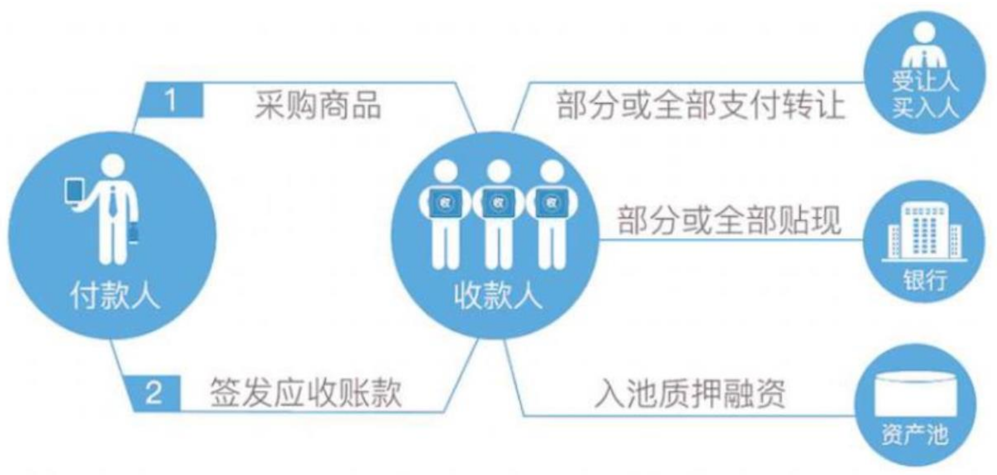


图 3 供应链应收账款资产流动

平台通过区块链连通供应链中的各方企业和金融机构，完整真实地记录资产（基于核心企业应付账款）的**上链**、**流通**、**拆分**和**兑付**。由于区块链上的数据经多方记录确认，不可篡改、不可抵赖、可以追溯，从而实现应收账款的拆分转让，并全部能够追溯至登记上链的初始资产。其中，在原始资产登记上链时，通过对供应商的应收账款进行审核校验与确权，确认贸易关系真实有效，以保证上链资产的真实可信，并实现核心企业对多级供应商的信用穿透。此外，平台还与多家金融机构进行合作，提升资金配置效率、支持小微企业基于供应链进行融资，降低融资成本，深度盘活金融资源。

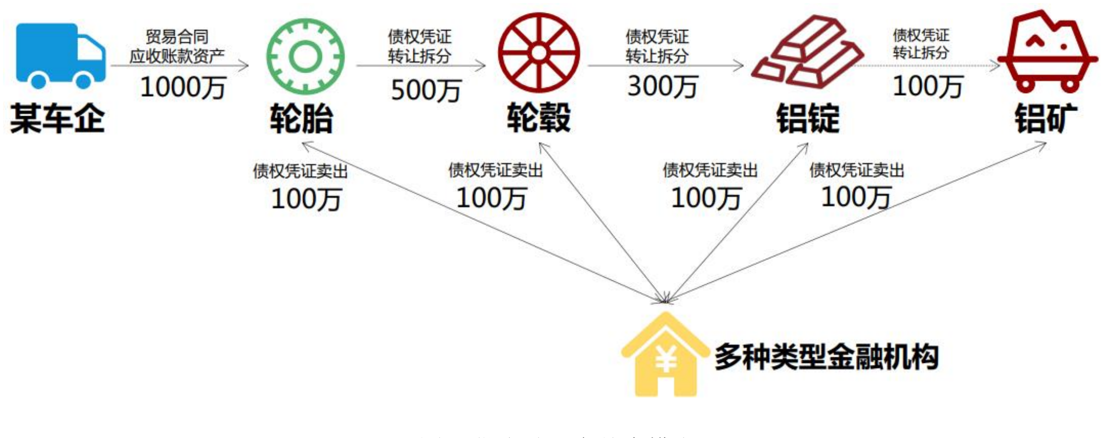


图 4 供应链平台基本模式

在实际操作中，将一级供应商（轮胎企业）与核心企业（某车企）之间的应收账款，通过资产网关进行全线上化电子审核，确保贸易背景真实性。核心企业对该笔应收账款进行确权后，进行数字化上链，形成数字债权凭证，后续可以将该凭证在供应链平台中进行拆分及转让。每一级供应商均可以按业务需要选择持有到期、融资卖出或转让来满足自己的资金诉求。

二、方案设计

2.1 数据结构说明

2.1.1 企业 Company

```
struct Company {
    string name;//公司名称
    address addr;//公司地址（唯一）
    int credit_rate;//公司信用级别
    uint oweAccount;//公司欠他人的总账款
    uint ownAccount;//公司拥有的总账款
    uint money;//公司在本链上拥有的总资金
    uint overdueFactor;//公司扣除信用分的系数
}
```

2.1.2 债款凭证 Voucher

```
struct Voucher {
    uint256 voucherID;//债款凭证 ID 号
    address loanTo;//本债款凭证的收款人
    address loanFrom;//本债款凭证的欠款人
    bool isValid;//本债款凭证的有效性
    uint amount;//本债款凭证的款数
    uint256 startDay;//本债款凭证的签发日期
}
```

2.1.3 全局变量说明

```
uint256 maxVoucherID;//当前最大有效债权凭证的 ID 号
uint256 minVoucherID;//当前最小有效债权凭证的 ID 号

uint256 payDeadline;//签发债权凭证后的兑付期限（只能由银行进行设置）

address public bank;//银行的地址

mapping(address => Company) companies;//公司的集合（通过地址唯一寻址公司）
mapping(uint256 => Voucher) vouchers;//债权凭证的集合（通过债权凭证的 ID 号唯一寻址凭证）
```

2.2 合约构造函数 constructor

```
constructor () public {
    bank = msg.sender;//由银行进行合约部署
    //将当前最大/最小有效债权凭证的 ID 号均初始化为0
    maxVoucherID = 0;
    minVoucherID = 0;
    //将银行作为公司进行初始化
    companies[bank].addr = bank;
    companies[bank].credit_rate = 100000;//合理设置银行信用分为一个较大值
    companies[bank].money = 1000000000;//合理设置银行初始拥有资金十亿
    companies[bank].oweAccount = 0;
    companies[bank].ownAccount = 0;
}
```

2.3 信用评分机制

本制品通过设计了一个信用评分机制对链上的注册公司进行信用评级：每个公司（除银行外）注册成立之初可获得**初始信用分** 800；**只有超过 5000 信用分（含 5000）的公司才有权限签发债权凭证**；另外采用**线性加分、乘性扣分**的方式对公司的信用积分进行累计，从而保证签发债权凭证的公司日后兑还债款的充分可能性，同时又保证供应链上的其他中小公司能够运转如故。

2.3.1 初始分数

每个公司（除银行外）注册成立之初均可获得 800 信用分的初始分数：

```
function incorporateCompany(string companyName) public  returns (address CompanyAddress, string CompanyName,
int CompanyCredit, uint CompanyOweAccount, uint CompanyOwnAccount, uint CompanyMoney) {
    address companyAddr = msg.sender;
    ...
    companies[companyAddr].credit_rate = 800;
    ...
}
```

2.3.2 签发债权凭证的要求

只有信用分大于 5000 分的公司才能签发债权凭证：

```
function issueDebtCertificate(uint amount, address seller) public returns (string CompanyName,
    uint CompanyOweAccount, int CompanyCredit, string SellerName, uint SellerOwnAccount, int SellerCredit) {
    /*check first*/
    address companyAddr = msg.sender;
    require(
        companies[companyAddr].credit_rate >= 5000,
        "You cannot issue a debt certificate."
    );
    ...
}
```

2.3.3 增加信用积分

公司可以通过四个途径提高信用积分：

✧ 使用资金进行交易，交易双方都将获得积分数：*交易额*/10^4

```
function payInCash(uint amount, address seller) public returns (string CompanyName,
int CompanyCredit, uint CompanyMoney, string SellerName, int SellerCredit, uint SellerMoney) {
    ...
    //add credit
    companies[companyAddr].credit_rate += (int)(amount / 10000);
    companies[seller].credit_rate += (int)(amount / 10000);
    ...
}
```

✧ 收到签发的债权凭证将获得积分数：*款额*/10^4

```
function issueDebtCertificate(uint amount, address seller) public returns (string CompanyName,
uint CompanyOweAccount, int CompanyCredit, string SellerName, uint SellerOwnAccount, int SellerCredit) {
    address companyAddr = msg.sender;
    ...
    //add seller's credit
    companies[seller].credit_rate += (int)(amount / 10000);
    ...
}
```

✧ 收到转让的债权凭证将获得积分数：*款额*/10^4

```
function payInAccount(uint amount, address seller) public returns (string CompanyName,
uint CompanyOwnAccount, string SellerName, uint SellerOwnAccount, int SellerCredit) {
    ...
    //add seller's credit
    companies[seller].credit_rate += (int)(amount / 10000);
    ...
}
```

✧ 使用资金兑付本司签发的债权凭证款额将获得积分数：*款额*/10^4

```
function payForDebts() public returns (address CompanyAddress, string CompanyName,
int CompanyCredit, uint CompanyOweAccount, uint CompanyOwnAccount, uint CompanyMoney) {
    address companyAddr = msg.sender;
    ...
    //add company's credit
    companies[companyAddr].credit_rate += (int)(companies[companyAddr].oweAccount / 10000);
    ...
}
```

2.3.4 扣减信用积分

若公司签发债权凭证后超过兑还期限还款将被扣减信用分数：$款额/10^4 \times 2^n$，其中n为公司过去逾期还款的次数，即公司第一次逾期还款时扣减信用分数$款额/10^4$，第二次则扣减$款额/10^4 \times 2$，以此类推。

```
function payForDebts() public returns (address CompanyAddress, string CompanyName,
int CompanyCredit, uint CompanyOweAccount, uint CompanyOwnAccount, uint CompanyMoney) {
    address companyAddr = msg.sender;
    ...
    bool isOverdue = false;
    //pay off
    for (uint256 i = minVoucherID + 1; i <= maxVoucherID; i ++) {
        if (vouchers[i].isValid == true && vouchers[i].loanFrom == companyAddr) {
            ...
            //check isOverdue
            if (now - vouchers[i].startDay * 24 * 3600 > payDeadline) {
                isOverdue = true;
            }
        }
    }
    //if overdue subtract company's credit
    if (isOverdue == true) {
        companies[companyAddr].credit_rate -= (int)(companies[companyAddr].oweAccount / 10000 *
companies[companyAddr].overdueFactor);
        companies[companyAddr].overdueFactor *= 2;
    }
    ...
}
```

2.4 公司的基本操作

2.4.1 在链上注册公司

公司可以在链上注册成立自己的公司，只有注册之后才能进行其他操作。

```
function incorporateCompany(string companyName) public returns (address CompanyAddress, string CompanyName,
int CompanyCredit, uint CompanyOweAccount, uint CompanyOwnAccount, uint CompanyMoney) {
    /*check first*/
    address companyAddr = msg.sender;
    require(
        companies[companyAddr].addr == 0,
        "Your company have incorporated, please don't do it again!"
    );
    /*body*/
    companies[companyAddr].addr = companyAddr;
    companies[companyAddr].name = companyName;
    companies[companyAddr].credit_rate = 800;
    companies[companyAddr].oweAccount = 0;
    companies[companyAddr].ownAccount = 0;
    companies[companyAddr].money = 0;
    companies[companyAddr].overdueFactor = 1;
}
```

2.4.2 转入资金

公司可以给自己的账户转入资金，转入的资金存在 money 中，可以用于交易或兑还债权凭证。

```
function rollIn(uint amount) public returns (address CompanyAddress, string CompanyName, int CompanyCredit,
uint CompanyOweAccount, uint CompanyOwnAccount, uint CompanyMoney) {
    address companyAddr = msg.sender;
    ...
    companies[companyAddr].money += amount;
    ...
}
```

2.4.3 转出资金

公司可以从自己的账户中提现资金，需注意提现金额不能超过公司账户拥有的资金。

```
function rollOut(uint amount) public returns (address CompanyAddress, string CompanyName, int CompanyCredit,
uint CompanyOweAccount, uint CompanyOwnAccount, uint CompanyMoney) {
    /*check first*/
    address companyAddr = msg.sender;
    require(
        companies[companyAddr].addr != 0,
        "You must incorporate your company first!"
    );
    require(
        companies[companyAddr].money >= amount,
        "You must have enough money to roll out."
    );
    /*body*/
    companies[companyAddr].money -= amount;
    ...
}
```

2.4.4 使用资金进行交易

公司可以直接使用自己账户所拥有的资金 money 进行交易，但需要注意的是买方公司拥有的资金必须大于或等于交易额。

```
function payInCash(uint amount, address seller) public returns (string CompanyName,
int CompanyCredit, uint CompanyMoney, string SellerName, int SellerCredit, uint SellerMoney) {
    /*check first*/
    address companyAddr = msg.sender;
    require(
        companies[companyAddr].addr != 0,
        "You must incorporate your company first!"
    );
    require(
        companies[seller].addr != 0,
        "The seller must have already incorporated!"
    );
    require(
        companies[companyAddr].money >= amount,
        "You must have enough money to pay in cash."
    );
    /*body*/
    //money transfer
    companies[companyAddr].money -= amount;
    companies[seller].money += amount;
    ...
}
```

2.5 有关债权凭证的操作

2.5.1 债权凭证的拆分与转让

公司拆分、转让债权凭证，换句话说，就是使用债权凭证进行交易，所以值得注意的是需要预先检查公司所拥有的总债权额是否大于或等于交易额，若小于则交易不能进行。

```
function payInAccount(uint amount, address seller) public returns (string CompanyName, uint CompanyOwnAccount,
string SellerName, uint SellerOwnAccount, int SellerCredit) {
    /*check first*/
    require(
        companies[msg.sender].addr != 0,
        "You must incorporate your company first!"
    );
    require(
        companies[seller].addr != 0,
        "The seller must have already incorporated!"
    );
    require(
        companies[msg.sender].ownAccount >= amount,
        "You must own enough debt certificate to pay."
    );
    /*body*/
    //get enough vouchers whose loanTo is msg.sender and is Valid is true
    uint tempAccount = 0;
    uint count = 0;
    for (uint256 i = minVoucherID + 1; i <= maxVoucherID; i++) {
        if (vouchers[i].loanTo == msg.sender && vouchers[i].isValid == true) {
            //temporary counters
            tempAccount += vouchers[i].amount;
            //old voucher processing
            vouchers[i].isValid = false;
            //add vouchers
            count += 1;
            vouchers[maxVoucherID + count].voucherID = maxVoucherID + count;
            vouchers[maxVoucherID + count].isValid = true;
            vouchers[maxVoucherID + count].startDay = vouchers[i].startDay;
            //transfer the debt certificate
            vouchers[maxVoucherID + count].loanTo = seller;
            vouchers[maxVoucherID + count].loanFrom = vouchers[i].loanFrom;
            if (tempAccount > amount) {
                vouchers[maxVoucherID + count].amount = vouchers[i].amount - (tempAccount - amount);
                //new another voucher
                count += 1;
                vouchers[maxVoucherID + count].voucherID = maxVoucherID + count;
                vouchers[maxVoucherID + count].loanTo = msg.sender;
                vouchers[maxVoucherID + count].loanFrom = vouchers[i].loanFrom;
                vouchers[maxVoucherID + count].isValid = true;
                vouchers[maxVoucherID + count].amount = tempAccount - amount;
                vouchers[maxVoucherID + count].startDay = vouchers[i].startDay;
                break;
            }
            else if (tempAccount == amount) {
                vouchers[maxVoucherID + count].amount = vouchers[i].amount;
                break;
            }
        }
    }
    ...
    //account transfer
    companies[msg.sender].ownAccount -= amount;
    companies[seller].ownAccount += amount;
    ...
}
```

2.5.2 使用债权凭证预先兑换资金

拥有一定债权额的公司可以在欠款公司兑还债款前到银行预先兑换资金，但需要注意的是公司拥有的债权额需要大于或等于公司想要兑换的资金数。

```
function accountForMoney(uint amount) public returns (address CompanyAddress, string CompanyName,
int CompanyCredit, uint CompanyOweAccount, uint CompanyOwnAccount, uint CompanyMoney) {
    /*check first*/
    address companyAddr = msg.sender;
    require(
        companies[companyAddr].addr != 0,
        "You must incorporate your company first!"
    );
    require(
        companies[companyAddr].ownAccount >= amount,
        "You must have enough account to exchange for money"
    );
    /*body*/
    //get enough vouchers whose loanTo is companyAddr and is Valid is true
    uint tempAccount = 0;
    uint count = 0;
    for (uint256 i = minVoucherID + 1; i <= maxVoucherID; i++) {
        if (vouchers[i].loanTo == companyAddr && vouchers[i].isValid == true) {
            //temporary counters
            tempAccount += vouchers[i].amount;
            //old voucher processing
            vouchers[i].isValid = false;
            //add vouchers
            count += 1;
            vouchers[maxVoucherID + count].voucherID = maxVoucherID + count;
            vouchers[maxVoucherID + count].isValid = true;
            vouchers[maxVoucherID + count].startDay = vouchers[i].startDay;
            //transfer the debt certificate to bank
            vouchers[maxVoucherID + count].loanTo = bank;
            vouchers[maxVoucherID + count].loanFrom = vouchers[i].loanFrom;
            if (tempAccount > amount) {
                vouchers[maxVoucherID + count].amount = vouchers[i].amount - (tempAccount - amount);
                count += 1;
                vouchers[maxVoucherID + count].voucherID = maxVoucherID + count;
                vouchers[maxVoucherID + count].loanTo = companyAddr;
                vouchers[maxVoucherID + count].loanFrom = vouchers[i].loanFrom;
                vouchers[maxVoucherID + count].isValid = true;
                vouchers[maxVoucherID + count].amount = tempAccount - amount;
                vouchers[maxVoucherID + count].startDay = vouchers[i].startDay;
                break;
            }
            else if (tempAccount == amount) {
                vouchers[maxVoucherID + count].amount = vouchers[i].amount;
                break;
            }
        }
    }
    ...
    //account transfer
    companies[companyAddr].ownAccount -= amount;
    companies[bank].ownAccount += amount;
    //add the company's money while subtract bank's money
    companies[companyAddr].money += amount;
    companies[bank].money -= amount;
    ...
}
```

2.5.3 公司兑还债权凭证

公司兑还债权凭证不仅要兑还所有欠款金额，还要付给银行一定的利息。在发起本交易以前，需要先检查公司是否有足够的金额一次付清所有的欠款金额以及给银行的利息。这里设置的是不允许公司超过一个月（30 天）兑还债款，否则银行可以采取其他诸如上诉等物理操作。

```
function payForDebts(uint t) public returns (address CompanyAddress, string CompanyName,
int CompanyCredit, uint CompanyOweAccount, uint CompanyOwnAccount, uint CompanyMoney) {
    /*check first*/
    address companyAddr = msg.sender;
    require(
        companies[companyAddr].addr != 0,
        "You must incorporate your company first!"
    );
    require(
        companies[companyAddr].addr != 0,
        "You must incorporate your company first!"
    );
    //compute the interest to bank
    uint maxInterest = companies[companyAddr].oweAccount * 30 / 10000;
    require(
        companies[companyAddr].money >= companies[companyAddr].oweAccount + maxInterest,
        "You must have enough money to pay your debts and interests off!"
    );
    /*body*/
    //add company's credit
    companies[companyAddr].credit_rate += (int)(companies[companyAddr].oweAccount / 10000);
    bool isOverdue = false;
    //pay off
    for (uint256 i = minVoucherID + 1; i <= maxVoucherID; i ++) {
        if (vouchers[i].isValid == true && vouchers[i].loanFrom == companyAddr) {
            vouchers[i].isValid = false;
            companies[companyAddr].money -= vouchers[i].amount;
            companies[companyAddr].oweAccount -= vouchers[i].amount;
            companies[vouchers[i].loanTo].money += vouchers[i].amount;
            companies[vouchers[i].loanTo].ownAccount -= vouchers[i].amount;
            //the interest to bank
            uint daysAfter = now / (24 * 3600) - vouchers[i].startDay;
            uint interest = vouchers[i].amount * daysAfter / 10000;
            companies[companyAddr].money -= interest;
            companies[bank].money += interest;
            ...
        }
    }
    ...
}
```

2.6 银行的操作

2.6.1 收取一定的利息

在这一供应链中，由于拥有债权额的公司可以提前到银行兑换资金，故而实际上由银行充当了最终的风险承担人。因此，本制品中通过让银行从债权凭证的签发者处获得一定的利息的方式来激励具有较大资金运转能力的银行承担风险。

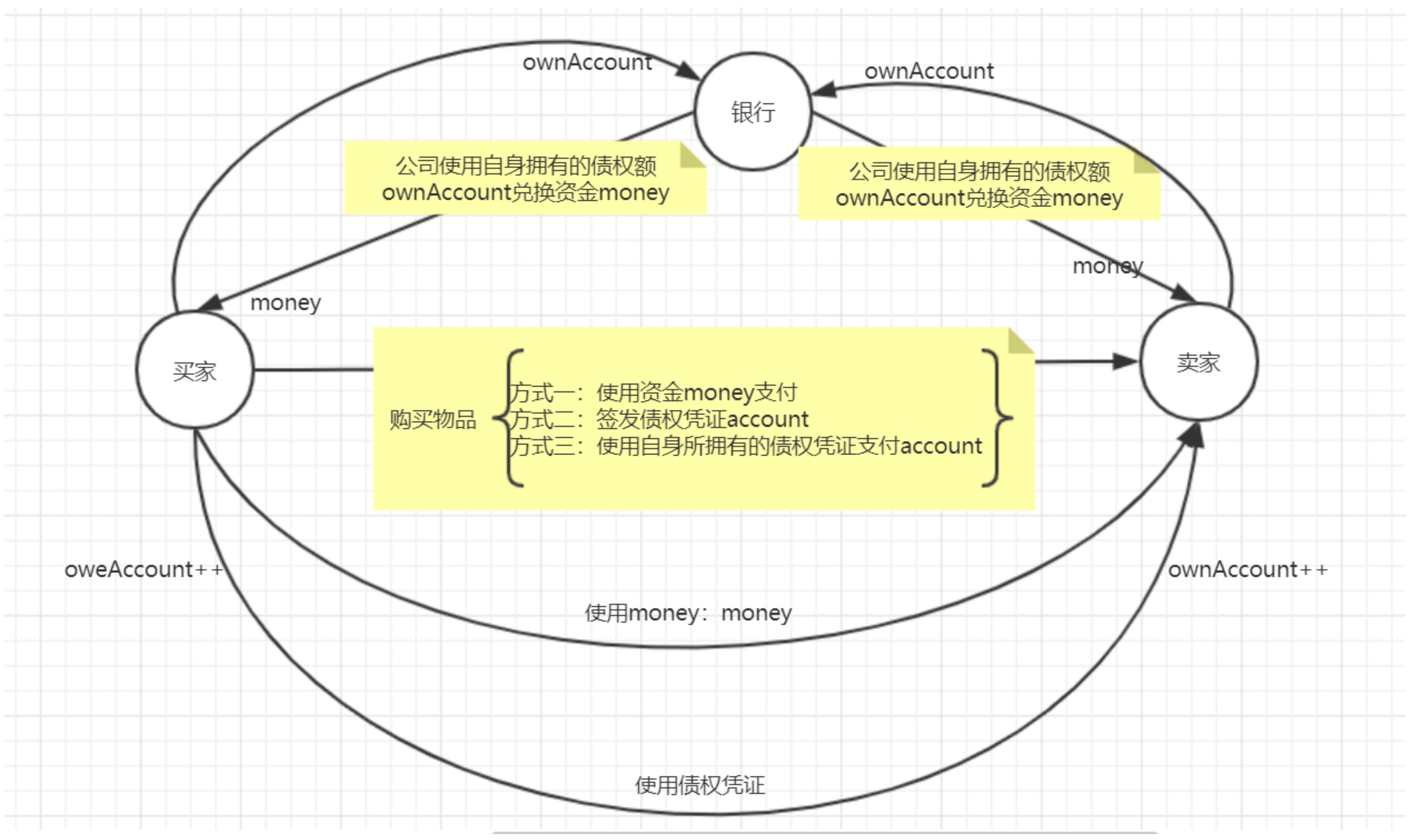
```
function payForDebts(uint t) public returns (address CompanyAddress, string CompanyName,
int CompanyCredit, uint CompanyOweAccount, uint CompanyOwnAccount, uint CompanyMoney) {
    address companyAddr = msg.sender;
    ...
    //pay off
    for (uint256 i = minVoucherID + 1; i <= maxVoucherID; i ++) {
        if (vouchers[i].isValid == true && vouchers[i].loanFrom == companyAddr) {
            ...
            //the interest to bank
            uint daysAfter = now / (24 * 3600) - vouchers[i].startDay;
            uint interest = vouchers[i].amount * daysAfter / 10000;
            companies[companyAddr].money -= interest;
            companies[bank].money += interest;
            ...
        }
    }
    ...
}
```

2.6.2 设置还款期限

银行可以设置债权凭证的还款期限（对于所有公司都一样），公司超过期限还款需要扣减信用分。此外，在本制品的限制中最大还款期限为 30 天，超过 30 天为不合法。

```
function setDeadline(uint deadline) public returns (uint DeadlineInSeconds){
    require(
        msg.sender == bank,
        "Only bank can call this function"
    );
    require(
        deadline < 30 * 1 days,
        "The max deadline is 30 days"
    );
    payDeadline = deadline;
    return payDeadline;
}
```

2.7 数据流图示


ownAccount
ownAccount
银行
公司使用自身拥有的债权额 ownAccount兑换资金money
公司使用自身拥有的债权额 ownAccount兑换资金money
money
money
买家
卖家
购买物品
方式一：使用资金money支付
方式二：签发债权凭证account
方式三：使用自身所拥有的债权凭证支付account
oweAccount++
ownAccount++
使用money：money
使用债权凭证

三、功能测试

3.0 测试用户

用户名称	用户公钥地址	用户说明
wyb_bank	0xb3d03b4e1a68a1bd2974b1fab6d41492ab4499ad	银行
web_company	0xa1d77f99132cc2f106af37f98d929e67b8f29910	供应链上的大公司
m_company	0x3eda1cf3d1e2e846b2052dac3ded4d413c046f31	供应链上的中小型公司
t_company	0x3eda1cf3d1e2e846b2052dac3ded4d413c046f31	供应链上的中小型公司
jj_company	0x9c1d7bc122dc317273c2e90916cbcb65b22164e8	供应链上的中小型公司

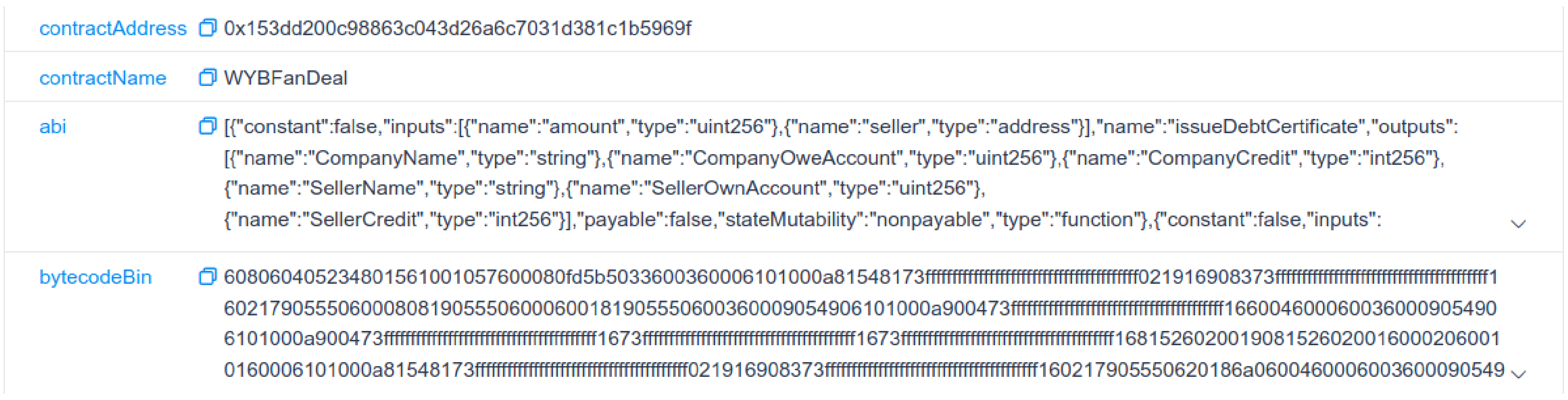
假设一个简单的供应链为：web_company ← m_company ← t_company ← jj_company，其中箭头表示的是产品原料的流动方向，资金流动方向则与之相反。

3.1 银行的操作

银行可以进行合约部署，还款期限设置操作。

3.1.1 合约部署

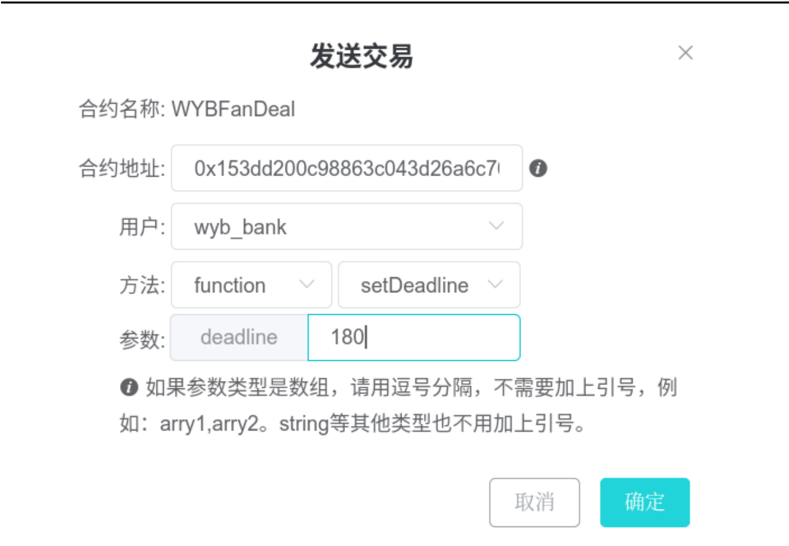
注意只能由银行进行合约部署，合约部署成功以后的返回结果：



contractAddress	0x153dd200c98863c043d26a6c7031d381c1b5969f
contractName	WYBFanDeal
abi	[{"constant":false,"inputs":[{"name":"amount","type":"uint256"},{"name":"seller","type":"address"}],"name":"issueDebtCertificate","outputs":[{"name":"CompanyName","type":"string"},{"name":"CompanyOweAccount","type":"uint256"},{"name":"CompanyCredit","type":"int256"},{"name":"SellerName","type":"string"},{"name":"SellerOwnAccount","type":"uint256"},{"name":"SellerCredit","type":"int256"}],"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":false,"inputs":
bytecodeBin	608060405234801561001057600080fd5b5033600360006101000a81548173ffffffffffffffffffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffffffffffffffff1602179055506000808190555060006001819055506003600090549061010000a900473ffffffffffffffffffffffffffffffffffffffff16600460006003600090549061010000a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff168152602001908152602001600020600101600061010000a81548173ffffffffffffffffffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffffffffffffffff16021790555062018a0600460006003600090549

3.1.2 设置还款期限

为了方便测试我们这里将还款期限设置为 3 分钟（180 秒）。



发送交易

合约名称: WYBFanDeal

合约地址: 0x153dd200c98863c043d26a6c7

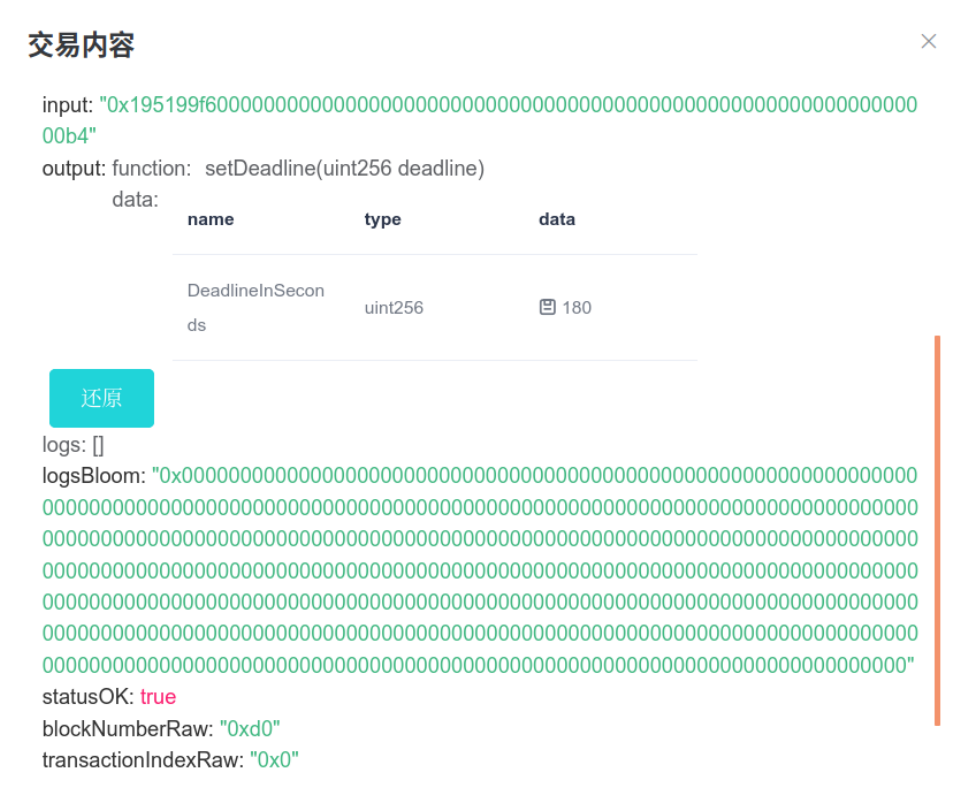
用户: wyb_bank

方法: function setDeadline

参数: deadline 180

如果参数类型是数组，请用逗号分隔，不需要加上引号，例如：arry1,arry2。string等其他类型也不用加上引号。

取消 确定



交易内容

```
input: "0x195199f6000000000000000000000000000000000000000000000000000000000000
00b4"
output: function:  setDeadline(uint256 deadline)
        data:
```

name	type	data
DeadlineInSeconds	uint256	180

还原

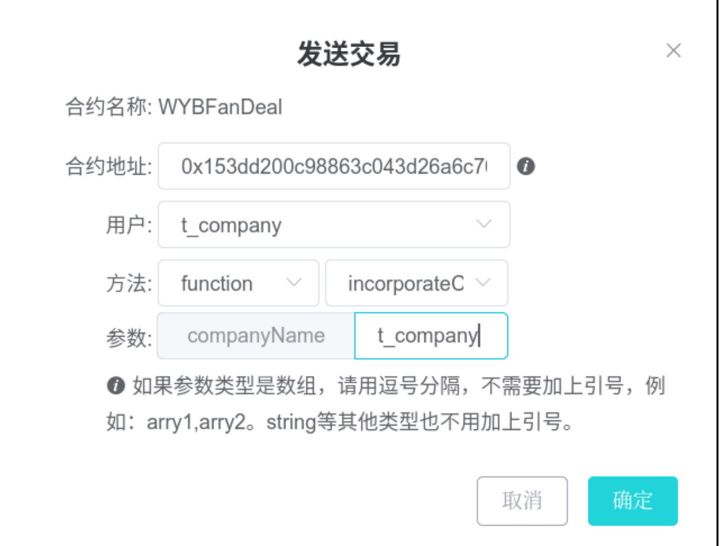
```
logs: []
logsBloom: "0x00000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000"
statusOK: true
blockNumberRaw: "0xd0"
transactionIndexRaw: "0x0"
```

目前所有公司签发债权凭证后的还款期限都设置为了 180 秒。

3.2 公司的操作

3.2.1 注册公司

以 t_company 为例，成功注册公司：



发送交易

合约名称: WYBFanDeal

合约地址: 0x153dd200c98863c043d26a6c7

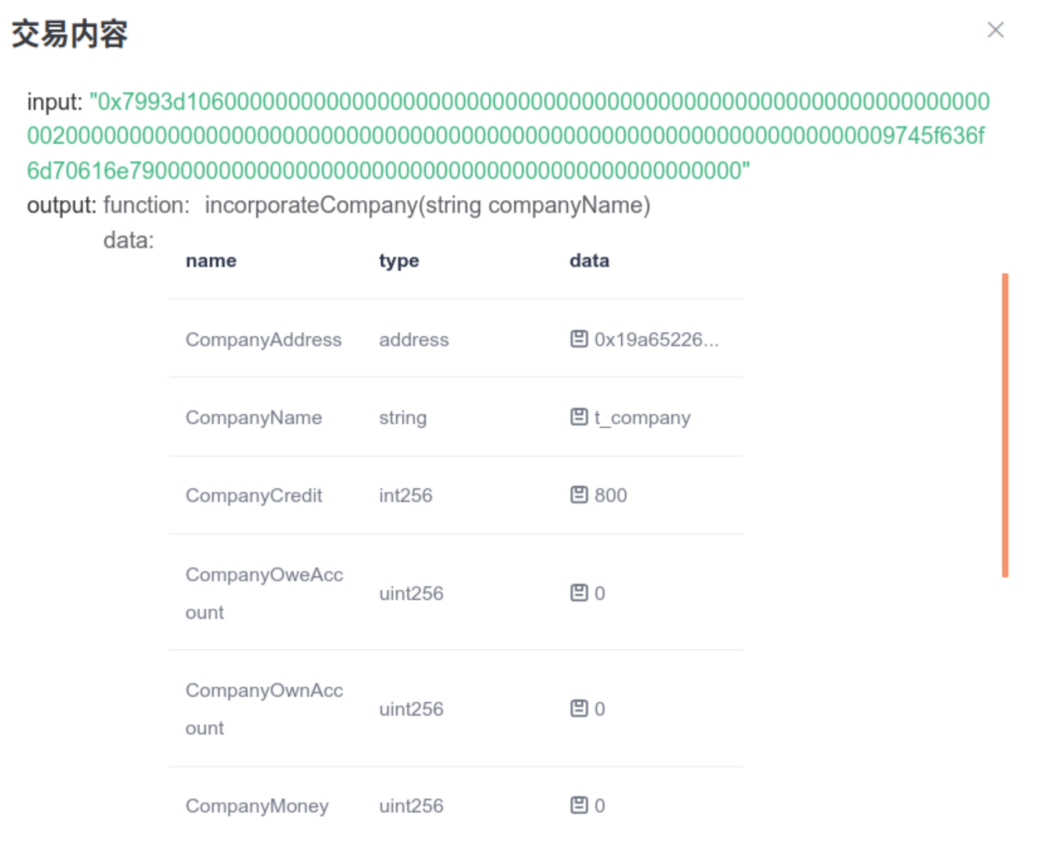
用户: t_company

方法: function incorporateC

参数: companyName t_company

如果参数类型是数组，请用逗号分隔，不需要加上引号，例如：arry1,arry2。string等其他类型也不用加上引号。

取消 确定

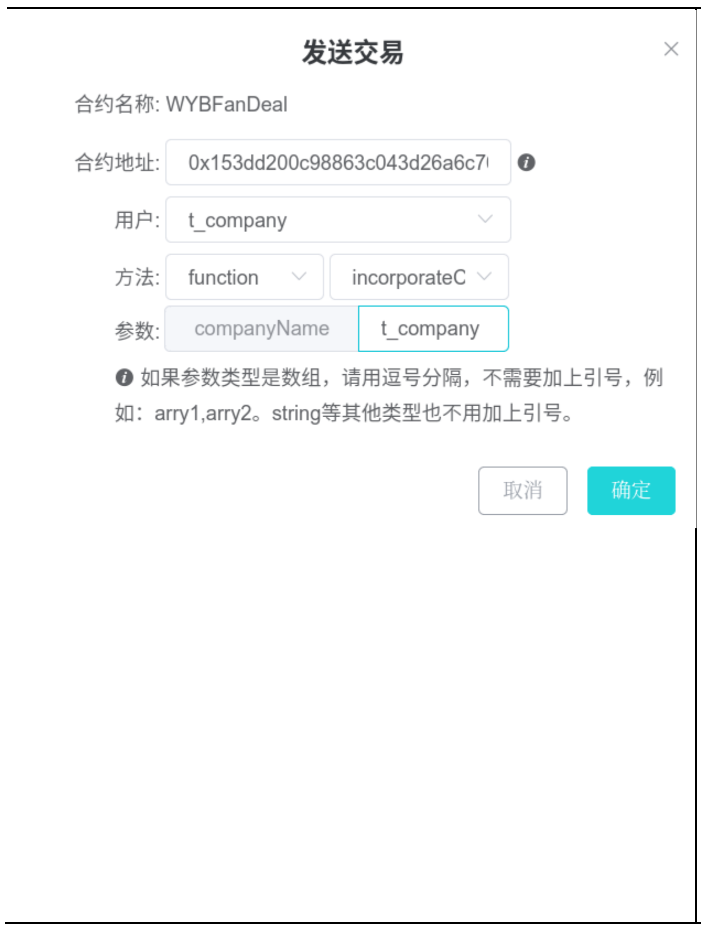


交易内容

```
input: "0x7993d106000000000000000000000000000000000000000000000000000000000000
0020000000000000000000000000000000000000000000000000000000000000009745f636f
6d70616e790000000000000000000000000000000000000000000000"
output: function:  incorporateCompany(string companyName)
        data:
```

name	type	data
CompanyAddress	address	0x19a65226...
CompanyName	string	t_company
CompanyCredit	int256	800
CompanyOweAccount	uint256	0
CompanyOwnAccount	uint256	0
CompanyMoney	uint256	0

若重复注册公司则交易不能执行：


发送交易

合约名称: WYBFanDeal

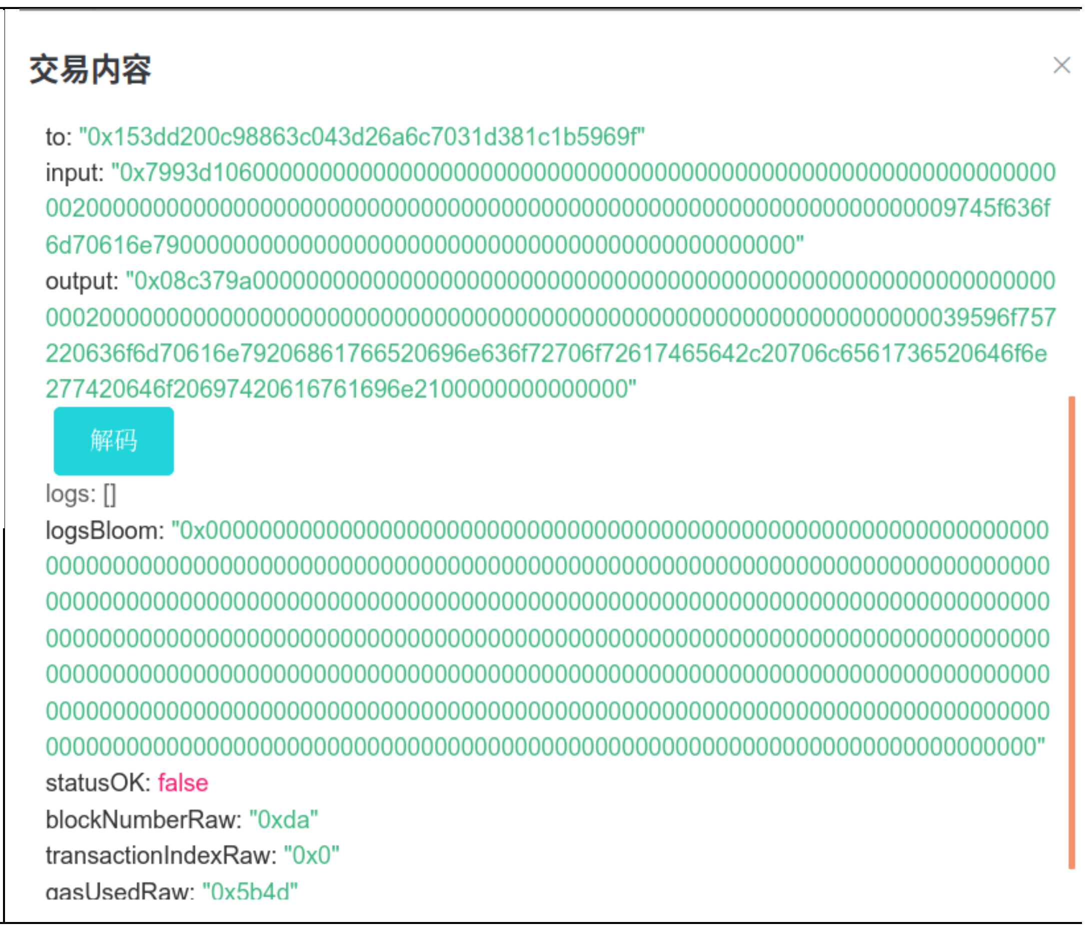
合约地址: 0x153dd200c98863c043d26a6c7(

用户: t_company

方法: function incorporateC

参数: companyName t_company

如果参数类型是数组，请用逗号分隔，不需要加上引号，例如：arry1,arry2。string等其他类型也不用加上引号。

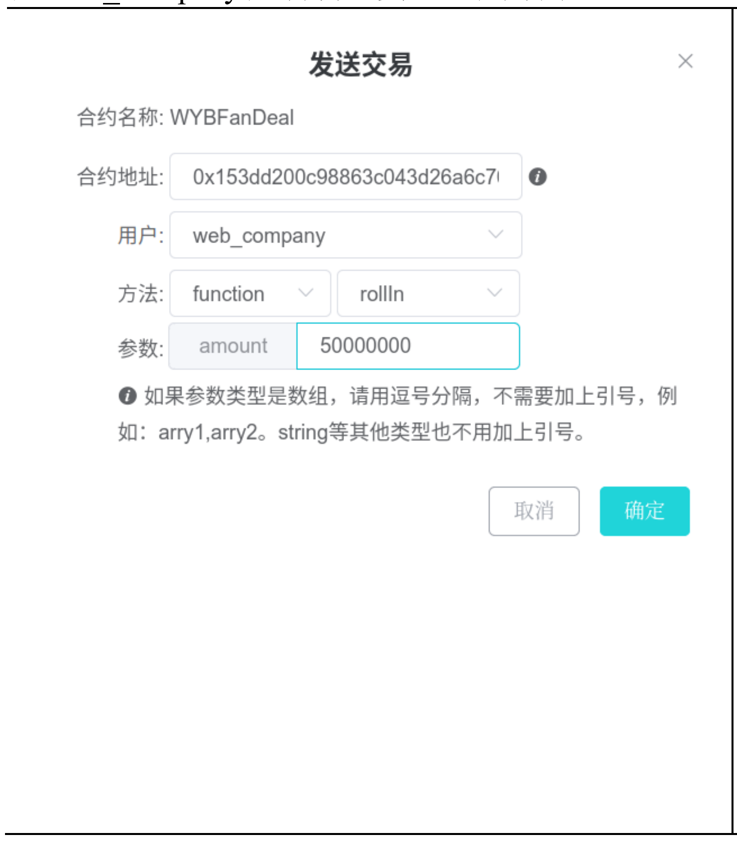

交易内容

```
to: "0x153dd200c98863c043d26a6c7031d381c1b5969f"
input: "0x7993d106000000000000000000000000000000000000000000000000000000000000
0020000000000000000000000000000000000000000000000000000000000000000009745f636f
6d70616e790000000000000000000000000000000000000000000000"
output: "0x08c379a000000000000000000000000000000000000000000000000000000000000
00020000000000000000000000000000000000000000000000000000000000000000039596f757
220636f6d70616e79206861766520696e636f72706f72617465642c20706c6561736520646f6e
277420646f20697420616761696e2100000000000000"
```

```
logs: []
logsBloom: "0x0000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000"
statusOK: false
blockNumberRaw: "0xda"
transactionIndexRaw: "0x0"
gasUsedRaw: "0x5b4d"
```

3.2.2 转入资金

以 web_company 成功转入资金 1 亿为例：


发送交易

合约名称: WYBFanDeal

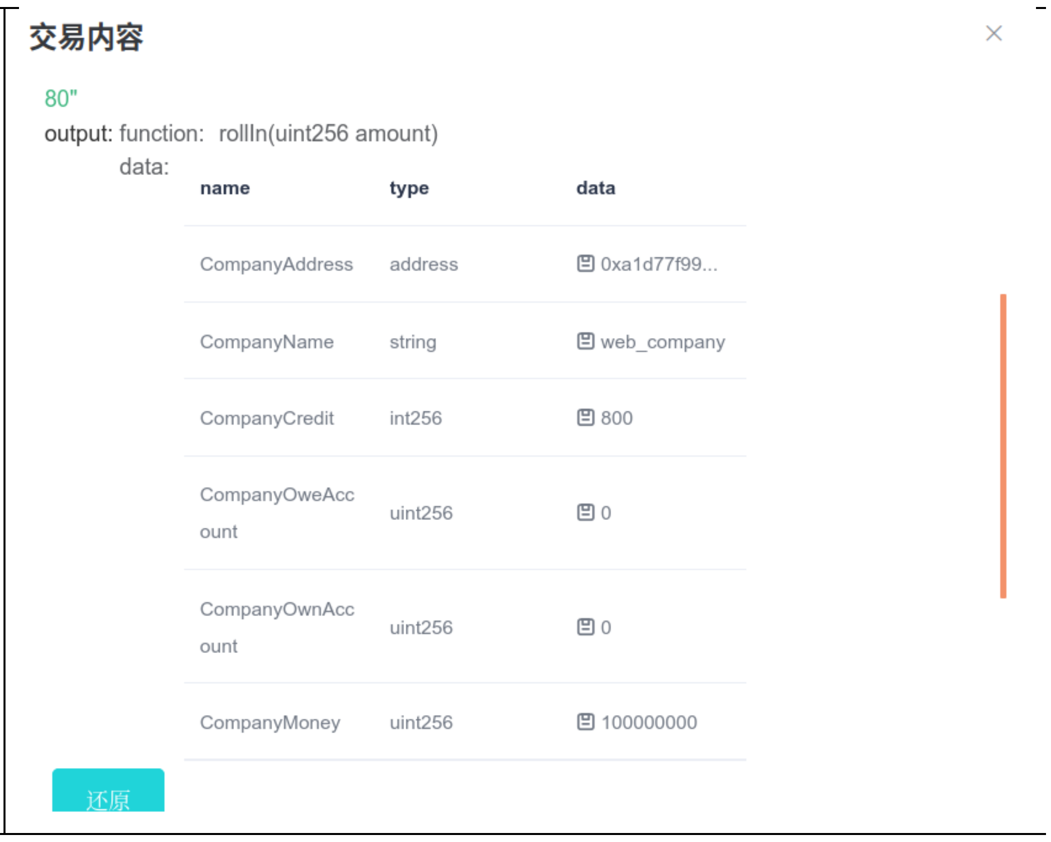
合约地址: 0x153dd200c98863c043d26a6c7(

用户: web_company

方法: function rollIn

参数: amount 50000000

如果参数类型是数组，请用逗号分隔，不需要加上引号，例如：arry1,arry2。string等其他类型也不用加上引号。


交易内容

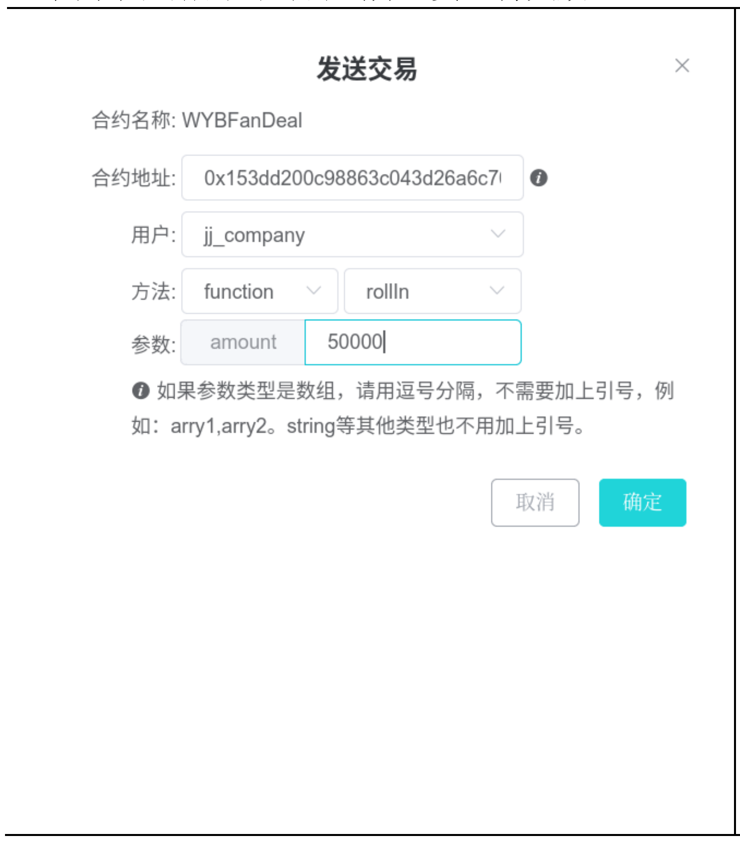
80"

output: function: rollIn(uint256 amount)

data:

name	type	data
CompanyAddress	address	0xa1d77f99...
CompanyName	string	web_company
CompanyCredit	int256	800
CompanyOweAccount	uint256	0
CompanyOwnAccount	uint256	0
CompanyMoney	uint256	100000000

一个尚未注册的公司尝试转入资金将失败：


发送交易

合约名称: WYBFanDeal

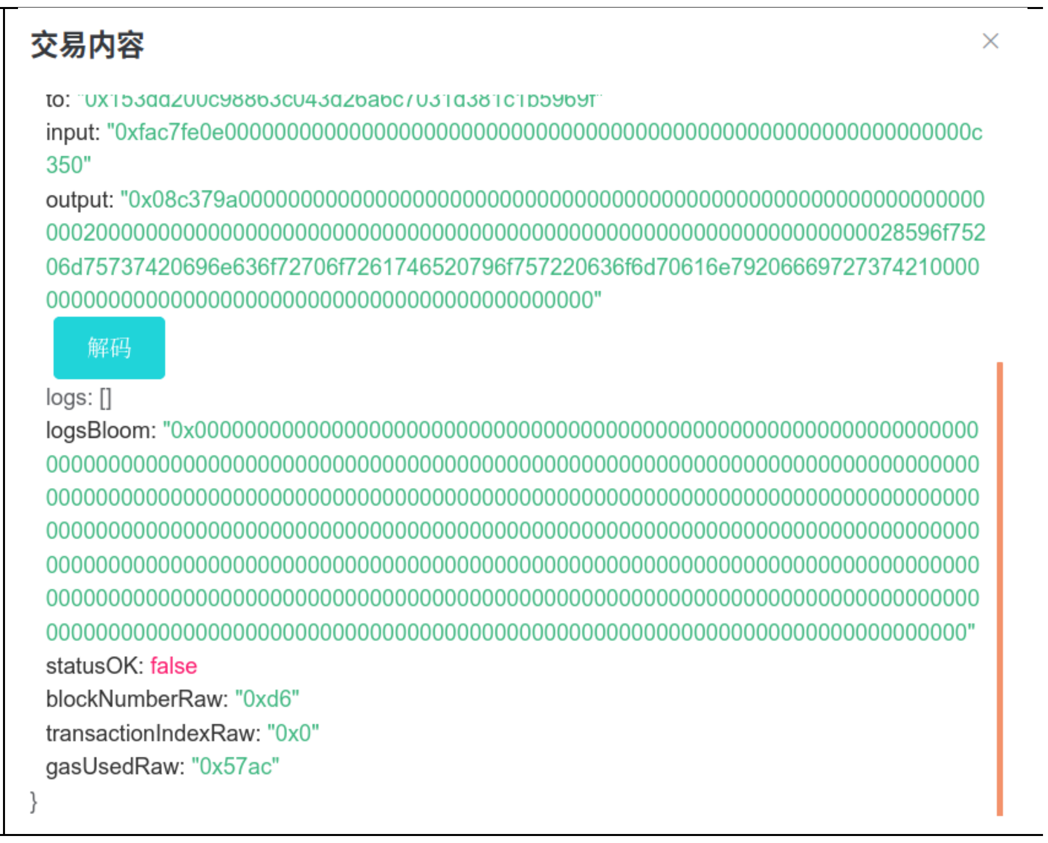
合约地址: 0x153dd200c98863c043d26a6c7(

用户: jj_company

方法: function rollIn

参数: amount 50000

如果参数类型是数组，请用逗号分隔，不需要加上引号，例如：arry1,arry2。string等其他类型也不用加上引号。

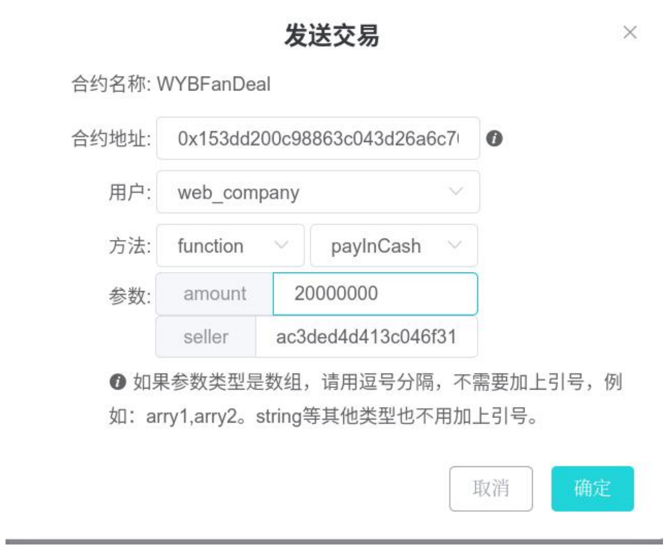

交易内容

```
to: "0x153dd200c98863c043d26a6c7031d381c1b5969f"
input: "0xfac7fe0e000000000000000000000000000000000000000000000000000000000000c
350"
output: "0x08c379a000000000000000000000000000000000000000000000000000000000000
00020000000000000000000000000000000000000000000000000000000000000000028596f752
06d75737420696e636f72706f7261746520796f757220636f6d70616e79206669727374210000
00000000000000000000000000000000000000000000"
```

```
logs: []
logsBloom: "0x0000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000"
statusOK: false
blockNumberRaw: "0xd6"
transactionIndexRaw: "0x0"
gasUsedRaw: "0x57ac"
}
```

3.2.3 使用资金进行交易

以 web_company 成功向 m_company 购买 2000 万的原料为例：

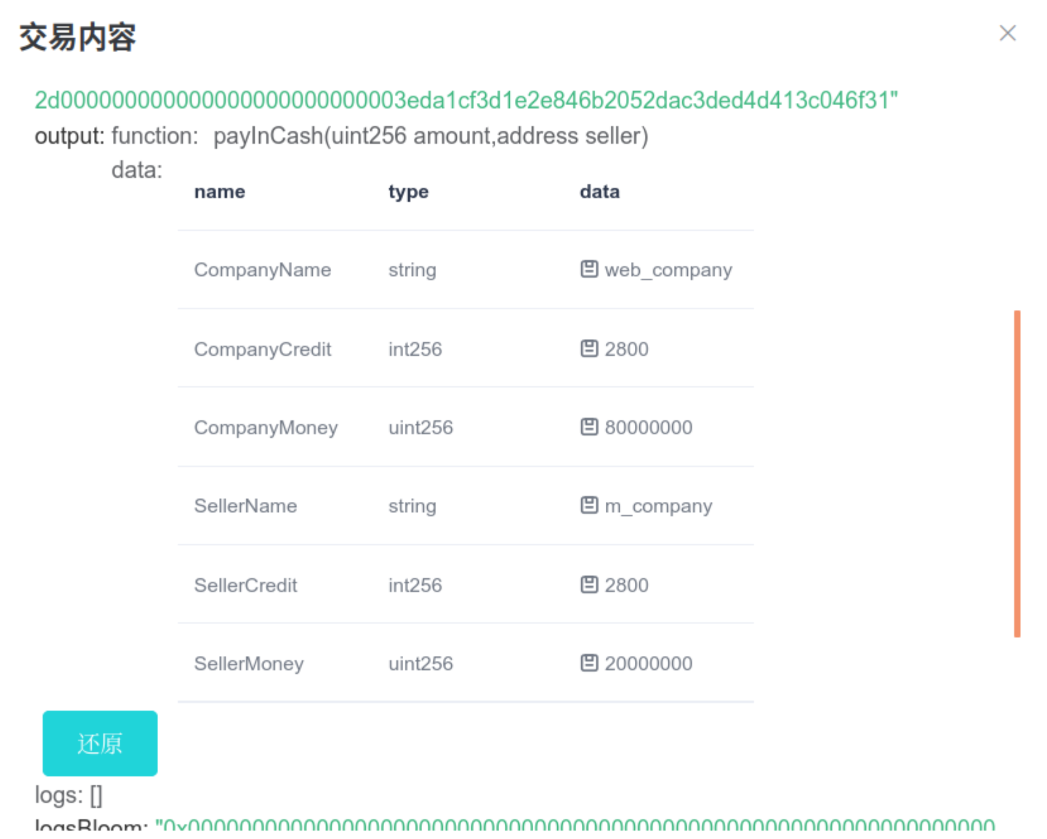


发送交易

合约名称: WYBFanDeal

合约地址:	0x153dd200c98863c043d26a6c7
用户:	web_company
方法:	function / payInCash
参数:	amount 20000000
	seller ac3ded4d413c046f31

如果参数类型是数组，请用逗号分隔，不需要加上引号，例如：arry1,arry2。string等其他类型也不用加上引号。



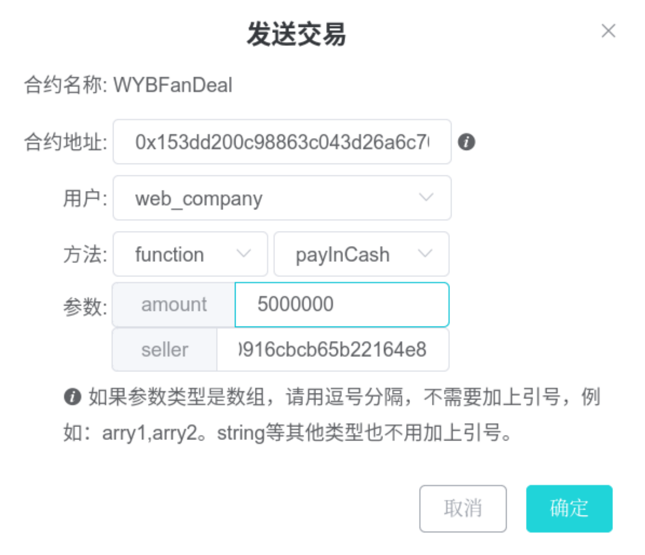
交易内容

```
2d000000000000000000000000003eda1cf3d1e2e846b2052dac3ded4d413c046f31"
output: function:  payInCash(uint256 amount,address seller)
        data:
```

name	type	data
CompanyName	string	web_company
CompanyCredit	int256	2800
CompanyMoney	uint256	80000000
SellerName	string	m_company
SellerCredit	int256	2800
SellerMoney	uint256	20000000

```
logs: []
logsBloom: "0x0000000000000000000000000000000000000000000000000000000000000000
```

若 web_company 向未注册公司（以 jj_company 为例）购买 500 万的原料，将失败：

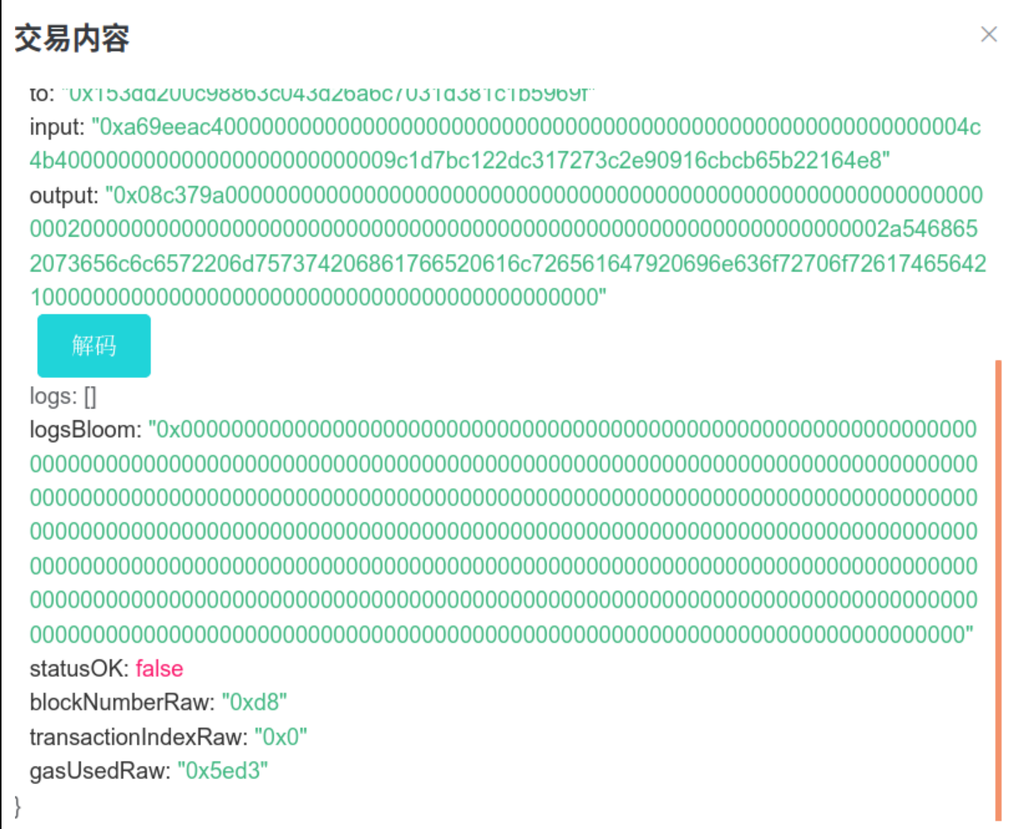


发送交易

合约名称: WYBFanDeal

合约地址:	0x153dd200c98863c043d26a6c7
用户:	web_company
方法:	function / payInCash
参数:	amount 5000000
	seller)916cbcb65b22164e8

如果参数类型是数组，请用逗号分隔，不需要加上引号，例如：arry1,arry2。string等其他类型也不用加上引号。

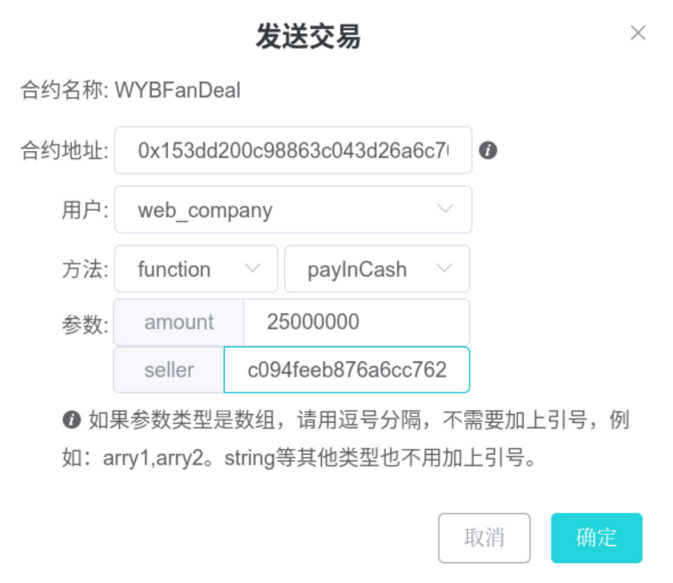


交易内容

```
to: "0x153dd200c98863c043d26a6c7031d381c1b5969f"
input: "0xa69eeac40000000000000000000000000000000000000000000000000000004c
4b4000000000000000000000000009c1d7bc122dc317273c2e90916cbcb65b22164e8"
output: "0x08c379a000000000000000000000000000000000000000000000000000000000
00020000000000000000000000000000000000000000000000000000000002a546865
2073656c6c6572206d75737420686176652061c7265616479206e636f72706f72617465642
1000000000000000000000000000000000000000000"
```

```
logs: []
logsBloom: "0x0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000"
statusOK: false
blockNumberRaw: "0xd8"
transactionIndexRaw: "0x0"
gasUsedRaw: "0x5ed3"
}
```

让 web_company 继续跟其他公司使用资金 money 交易以获得更多的信用积分（以 web_company 向 t_company 购买 2500 万的产品原料为例）：

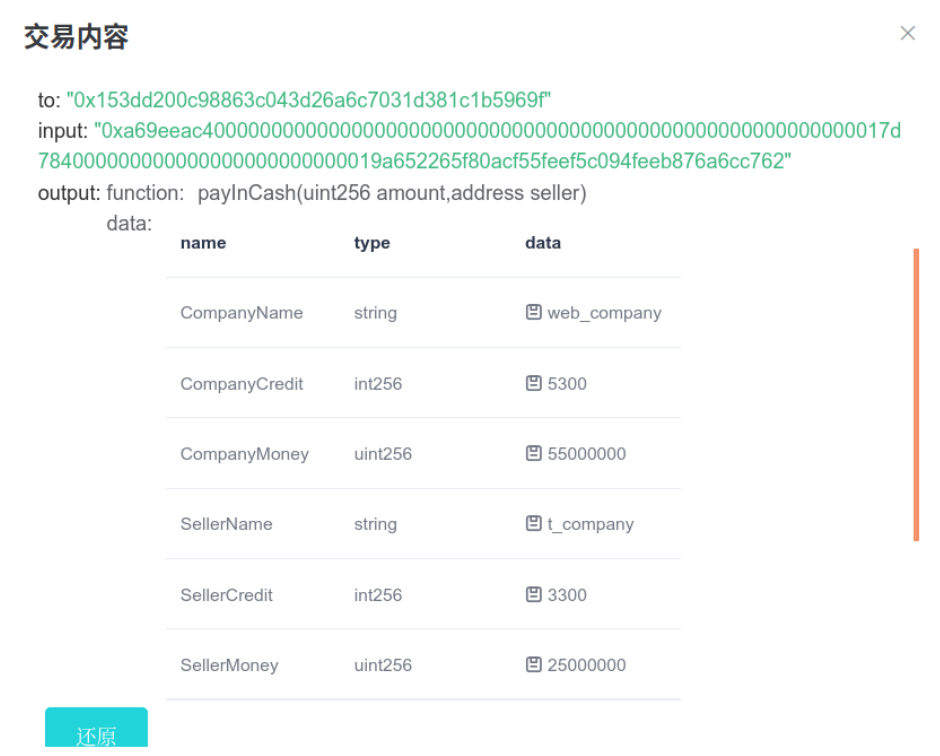


发送交易

合约名称: WYBFanDeal

合约地址:	0x153dd200c98863c043d26a6c7
用户:	web_company
方法:	function / payInCash
参数:	amount 25000000
	seller c094feeb876a6cc762

如果参数类型是数组，请用逗号分隔，不需要加上引号，例如：arry1,arry2。string等其他类型也不用加上引号。



交易内容

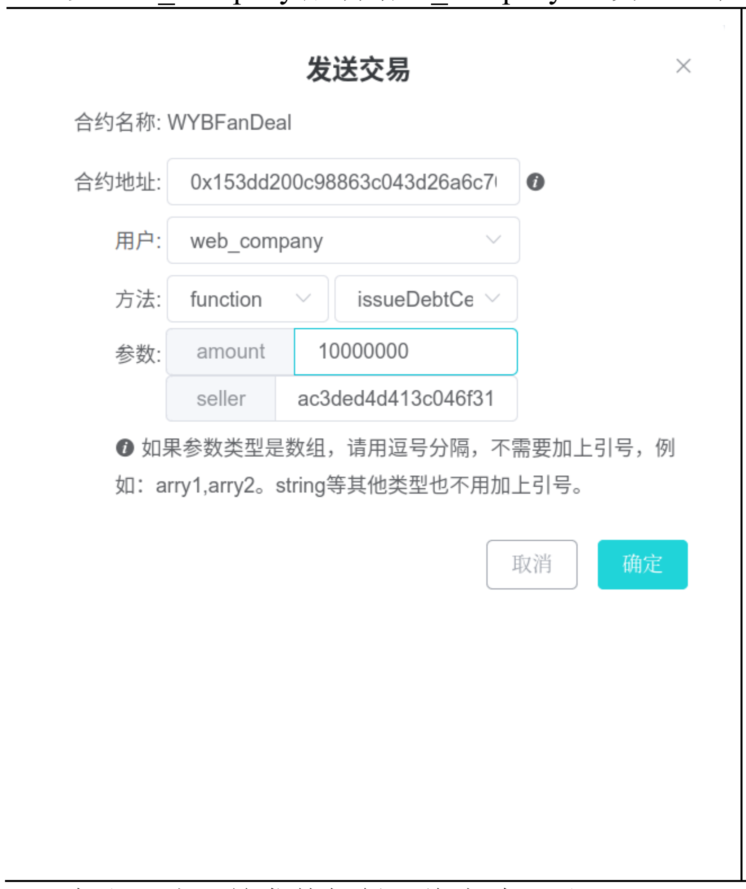
```
to: "0x153dd200c98863c043d26a6c7031d381c1b5969f"
input: "0xa69eeac400000000000000000000000000000000000000000000000000000017d
78400000000000000000000000000019a652265f80acf55feef5c094feeb876a6cc762"
output: function:  payInCash(uint256 amount,address seller)
        data:
```

name	type	data
CompanyName	string	web_company
CompanyCredit	int256	5300
CompanyMoney	uint256	55000000
SellerName	string	t_company
SellerCredit	int256	3300
SellerMoney	uint256	25000000

现在 web_company 就拥有了大于 5000 的信用积分，可以签发债权凭证了。

3.2.4 签发债权凭证

以 web_company 成功给 m_company 签发 1000 万的债权凭证为例：

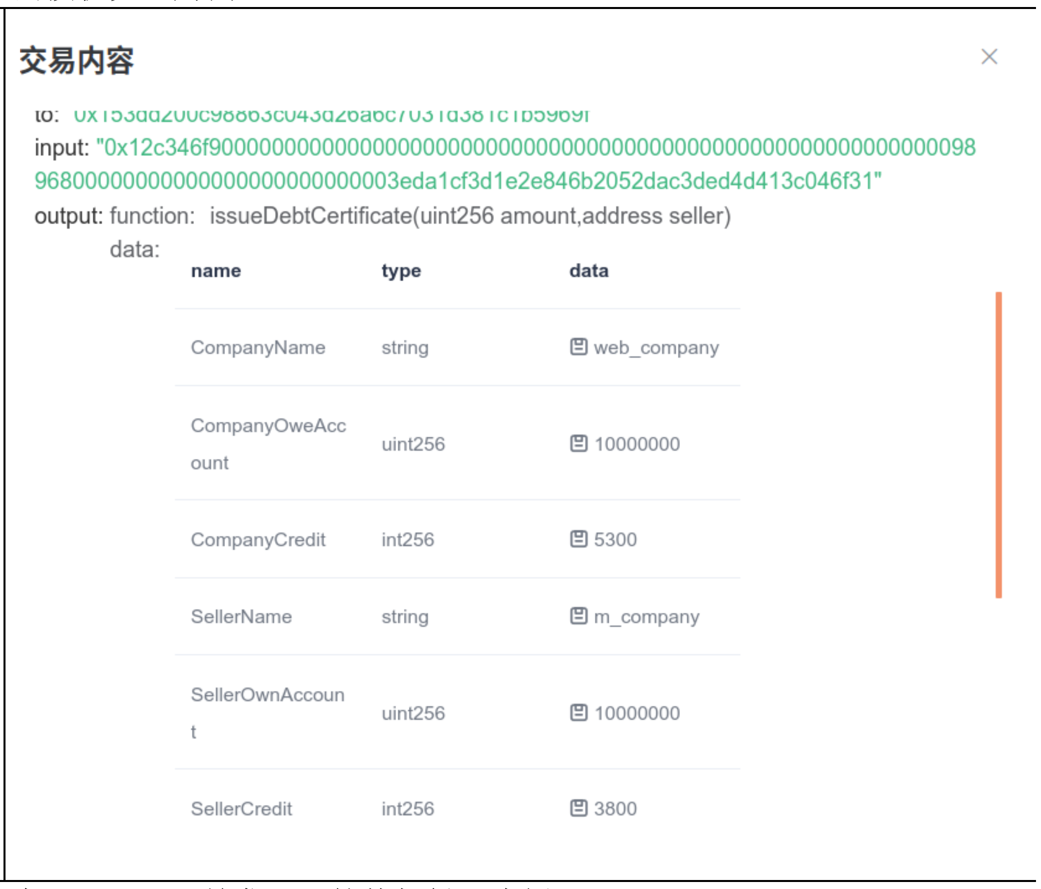


发送交易

合约名称: WYBFanDeal

合约地址:	0x153dd200c98863c043d26a6c7
用户:	web_company
方法:	function / issueDebtCe
参数:	amount 10000000
	seller ac3ded4d413c046f31

如果参数类型是数组，请用逗号分隔，不需要加上引号，例如：arry1,arry2。string等其他类型也不用加上引号。

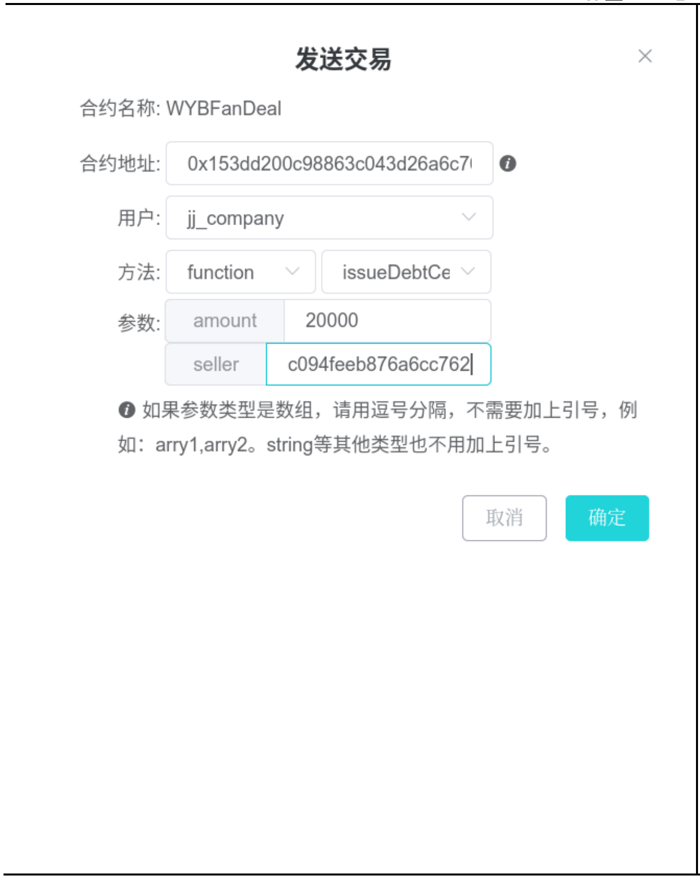


交易内容

```
to: "0x153dd200c98863c043d26a6c7031d381c1b5969f"
input: "0x12c346f900000000000000000000000000000000000000000000000000000098
96800000000000000000000000003eda1cf3d1e2e846b2052dac3ded4d413c046f31"
output: function:  issueDebtCertificate(uint256 amount,address seller)
        data:
```

name	type	data
CompanyName	string	web_company
CompanyOweAccount	uint256	10000000
CompanyCredit	int256	5300
SellerName	string	m_company
SellerOwnAccount	uint256	10000000
SellerCredit	int256	3800

未注册公司签发债权凭证将失败（以 jj_company 向 t_company 签发 2 万的债权凭证为例）：

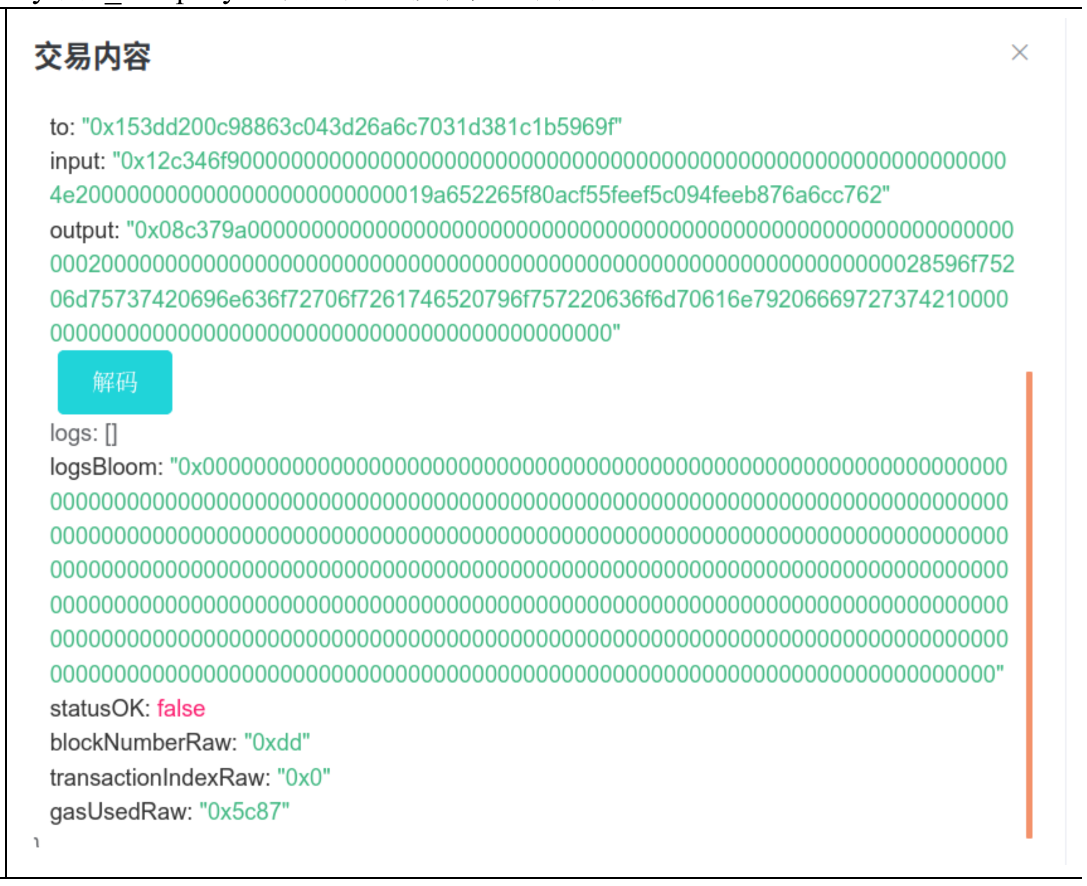


发送交易

合约名称: WYBFanDeal

合约地址:	0x153dd200c98863c043d26a6c7
用户:	jj_company
方法:	function / issueDebtCe
参数:	amount 20000
	seller c094feeb876a6cc762

如果参数类型是数组，请用逗号分隔，不需要加上引号，例如：arry1,arry2。string等其他类型也不用加上引号。

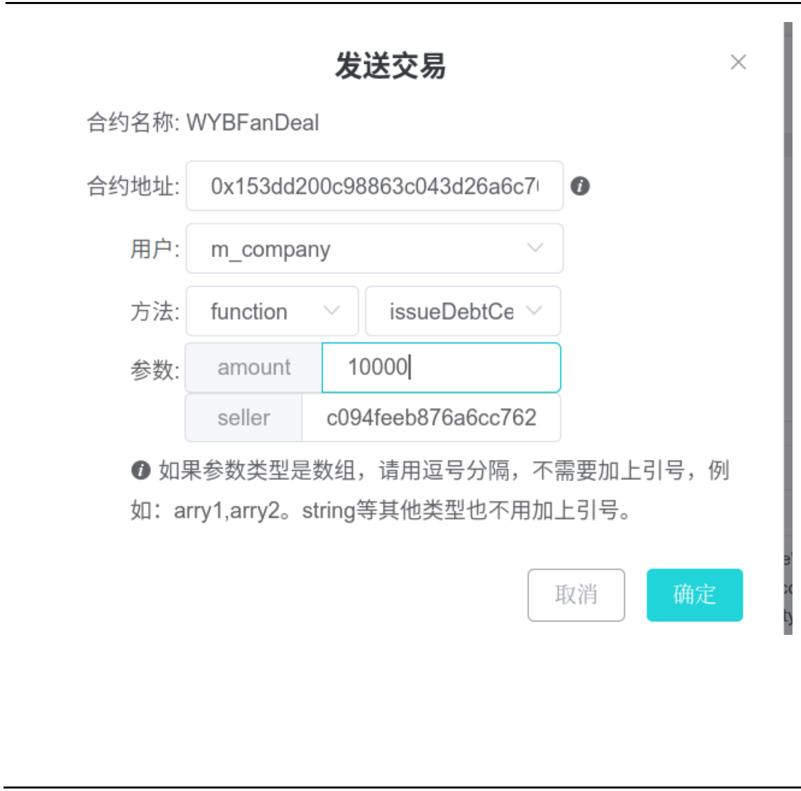


交易内容

```
to: "0x153dd200c98863c043d26a6c7031d381c1b5969f"
input: "0x12c346f900000000000000000000000000000000000000000000000000000000
4e20000000000000000000000000019a652265f80acf55feef5c094feeb876a6cc762"
output: "0x08c379a000000000000000000000000000000000000000000000000000000000
00020000000000000000000000000000000000000000000000000000000000028596f752
06d75737420696e636f72706f7261746520796f757220636f6d70616e79206669727374210000
000000000000000000000000000000000000000000"
```

```
logs: []
logsBloom: "0x000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000"
statusOK: false
blockNumberRaw: "0xdd"
transactionIndexRaw: "0x0"
gasUsedRaw: "0x5c87"
```

信用分未达到 5000 分及以上，签发债权凭证将失败（以 m_company 向 t_company 签发 1 万的债权凭证为例）：

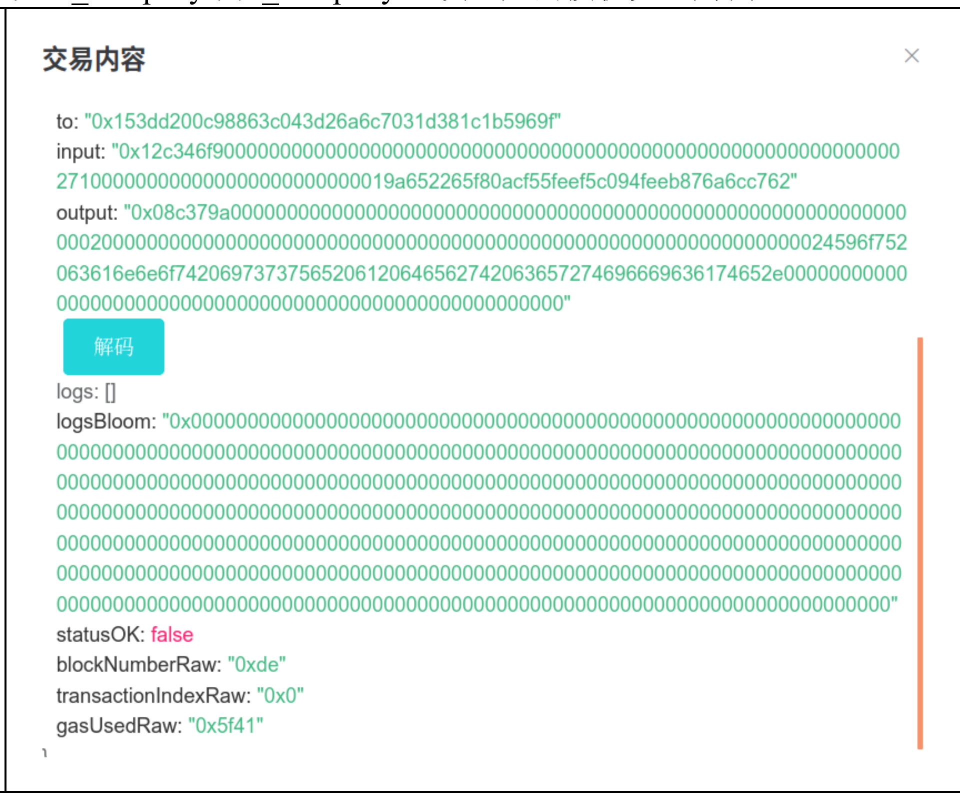


发送交易

合约名称: WYBFanDeal

合约地址:	0x153dd200c98863c043d26a6c7
用户:	m_company
方法:	function / issueDebtCe
参数:	amount 10000
	seller c094feeb876a6cc762

如果参数类型是数组，请用逗号分隔，不需要加上引号，例如：arry1,arry2。string等其他类型也不用加上引号。

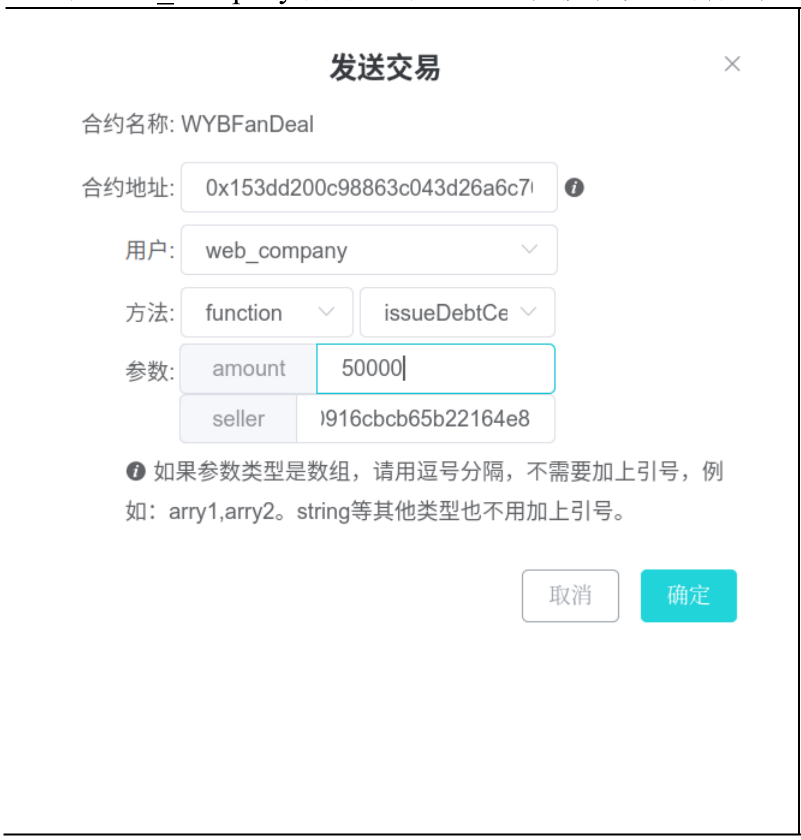


交易内容

```
to: "0x153dd200c98863c043d26a6c7031d381c1b5969f"
input: "0x12c346f900000000000000000000000000000000000000000000000000000000
2710000000000000000000000000019a652265f80acf55feef5c094feeb876a6cc762"
output: "0x08c379a000000000000000000000000000000000000000000000000000000000
00020000000000000000000000000000000000000000000000000000000000024596f752
063616e6e6f742069737375652061206465627420636572746966696361746552e000000000
000000000000000000000000000000000000000"
```

```
logs: []
logsBloom: "0x000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000"
statusOK: false
blockNumberRaw: "0xde"
transactionIndexRaw: "0x0"
gasUsedRaw: "0x5f41"
```

若 web_company 向未注册公司签发债权凭证将失败（以 web_company 向 jj_company 签发 5 万的债权凭证为例）：


发送交易

合约名称: WYBFanDeal

合约地址: 0x153dd200c98863c043d26a6c7

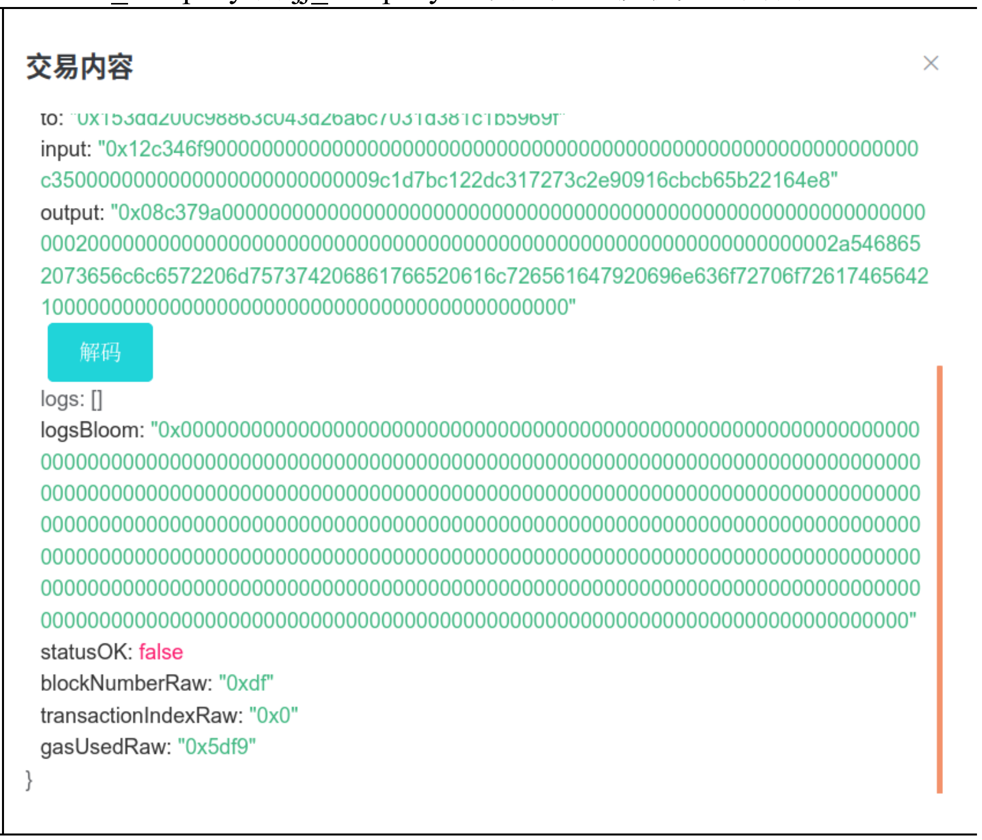
用户: web_company

方法: function issueDebtCe

参数: amount 50000

seller)916cbcb65b22164e8

如果参数类型是数组，请用逗号分隔，不需要加上引号，例如：arry1,arry2。string等其他类型也不用加上引号。


交易内容

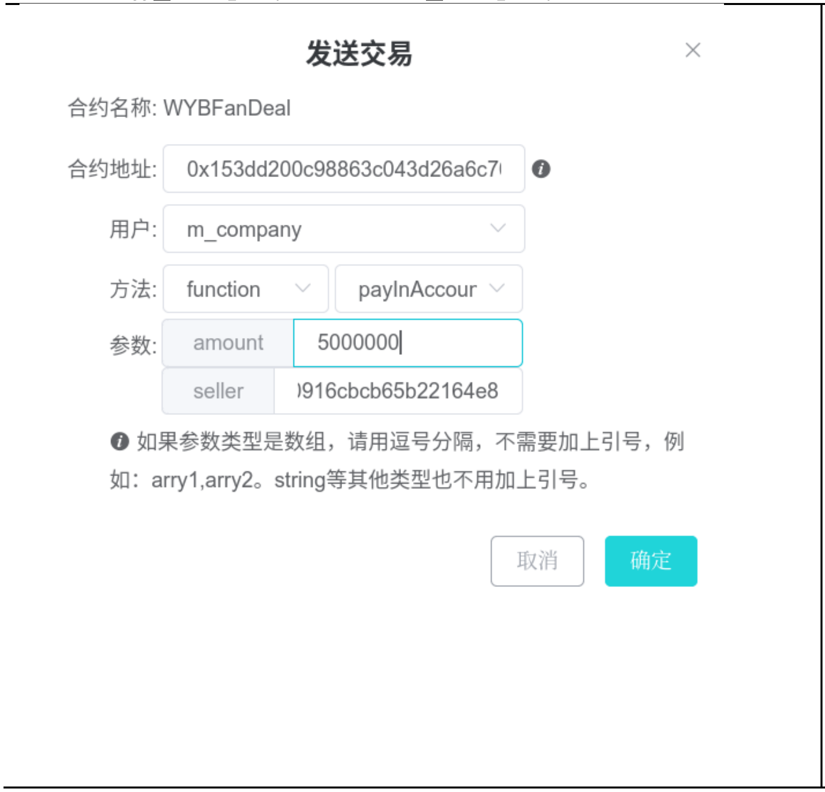
```
to: "0x153dd200c98863c043d26a6c7031d381c1b5969f"
input: "0x12c346f9000000000000000000000000000000000000000000000000000000000000
c35000000000000000000000000009c1d7bc122dc317273c2e90916cbcb65b22164e8"
output: "0x08c379a000000000000000000000000000000000000000000000000000000000000
00020000000000000000000000000000000000000000000000000000000000000002a546865
2073656c6c6572206d757374206861766520616c726561647920696e636f72706f72617465642
1000000000000000000000000000000000000000000"
```

```
logs: []
logsBloom: "0x0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000"
statusOK: false
blockNumberRaw: "0xdf"
transactionIndexRaw: "0x0"
gasUsedRaw: "0x5df9"
}
```

未注册公司不能执行任何操作，有关其错误操作均与上述测试类似，为了方便起见，以下将注册 jj_company，不再进行相应的错误测试。

3.2.5 使用债权凭证进行交易

注册 jj_company 后，以 m_company 成功使用债权额向 jj_company 购买 500 万的产品原料为例：


发送交易

合约名称: WYBFanDeal

合约地址: 0x153dd200c98863c043d26a6c7

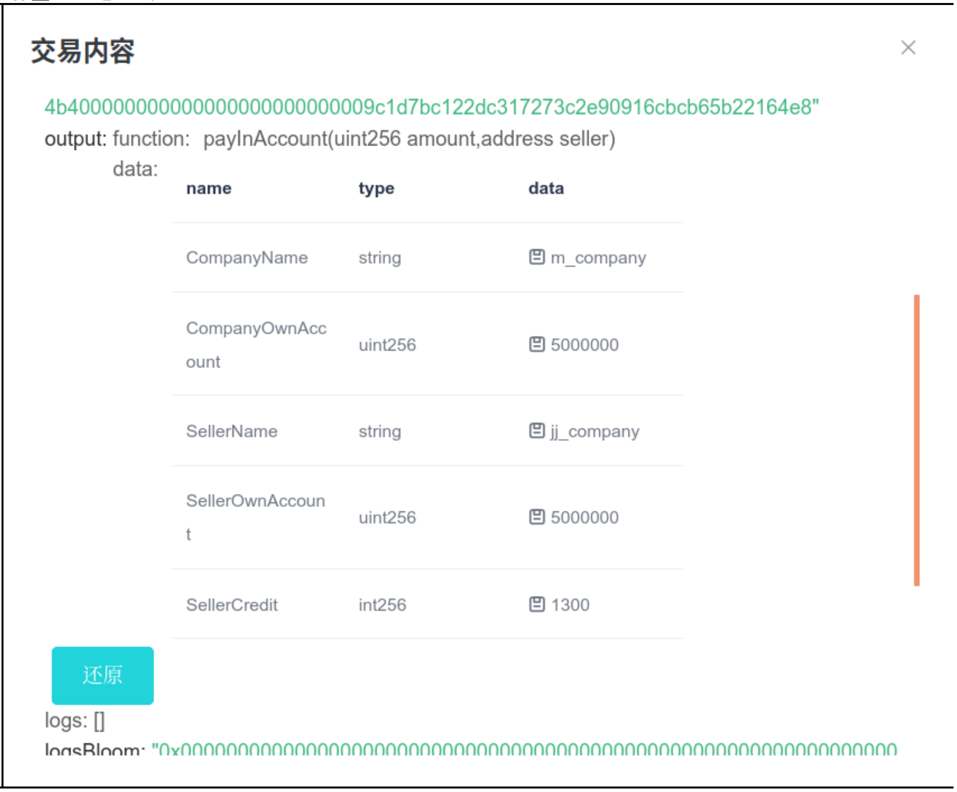
用户: m_company

方法: function payInAccour

参数: amount 5000000

seller)916cbcb65b22164e8

如果参数类型是数组，请用逗号分隔，不需要加上引号，例如：arry1,arry2。string等其他类型也不用加上引号。

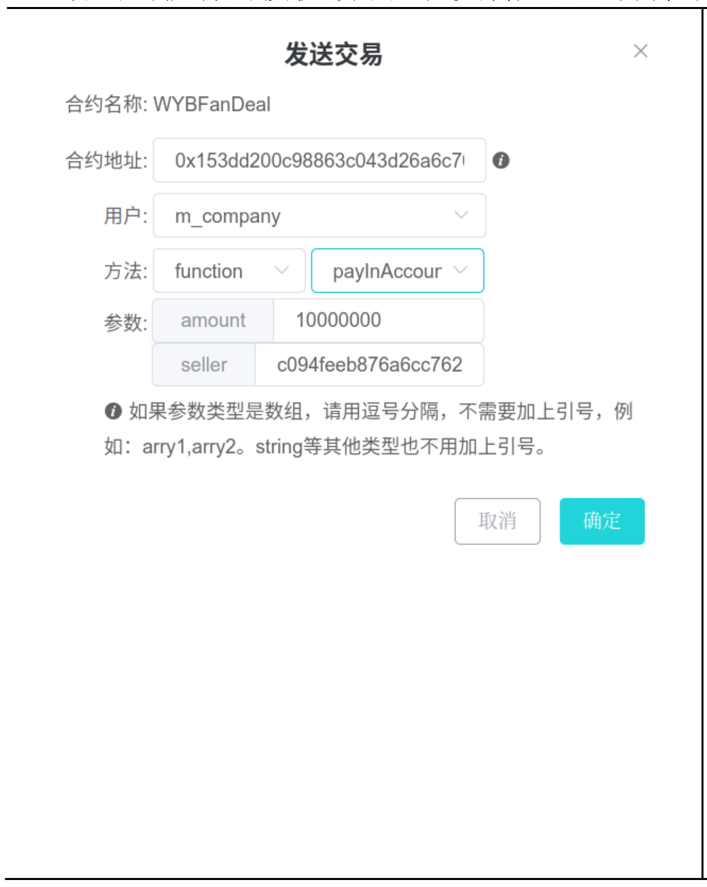

交易内容

```
4b40000000000000000000000009c1d7bc122dc317273c2e90916cbcb65b22164e8"
output: function:  payInAccount(uint256 amount,address seller)
         data:
```

name	type	data
CompanyName	string	m_company
CompanyOwnAccount	uint256	5000000
SellerName	string	jj_company
SellerOwnAccount	uint256	5000000
SellerCredit	int256	1300

```
logs: []
logsBloom: "0x000000000000000000000000000000000000000000000000000000000000000
```

若公司拥有的债权额不足以支付产品原料将导致交易失败(以 m_company 尝试向 t_company 购买 1000 万的产品原料为例)：


发送交易

合约名称: WYBFanDeal

合约地址: 0x153dd200c98863c043d26a6c7

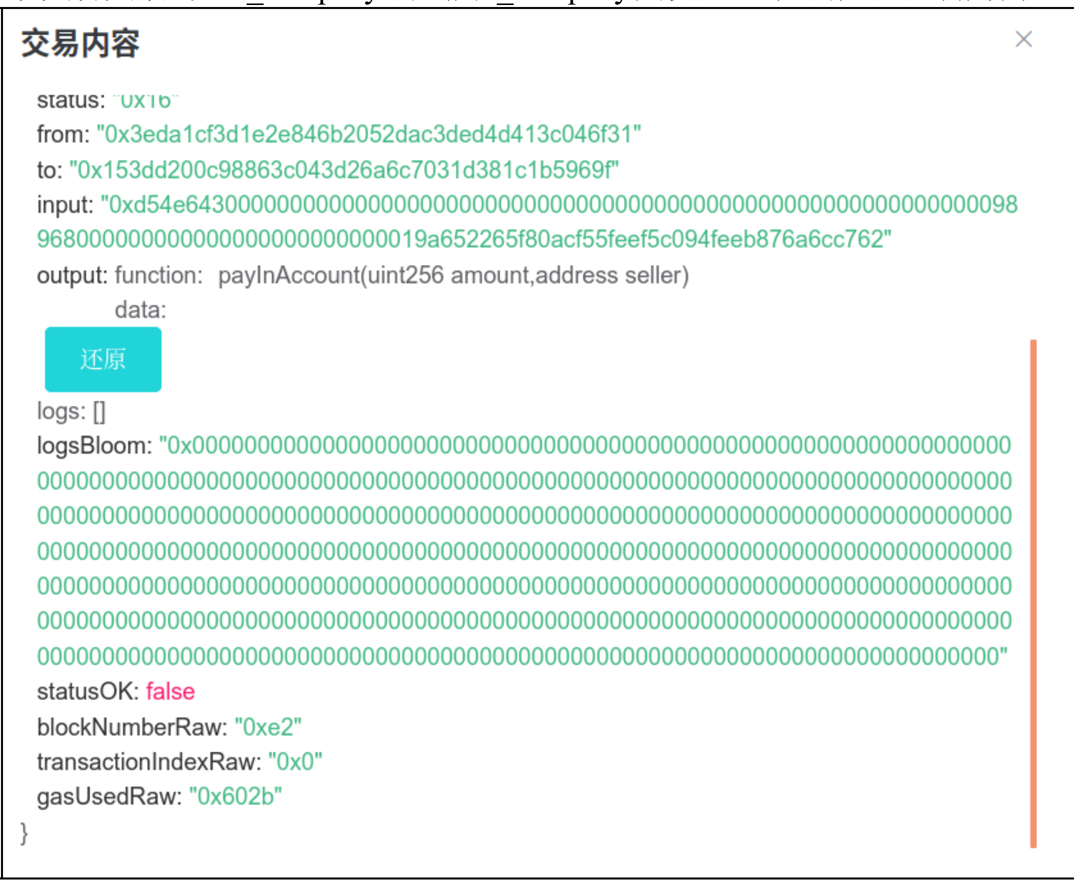
用户: m_company

方法: function payInAccour

参数: amount 10000000

seller c094feeb876a6cc762

如果参数类型是数组，请用逗号分隔，不需要加上引号，例如：arry1,arry2。string等其他类型也不用加上引号。

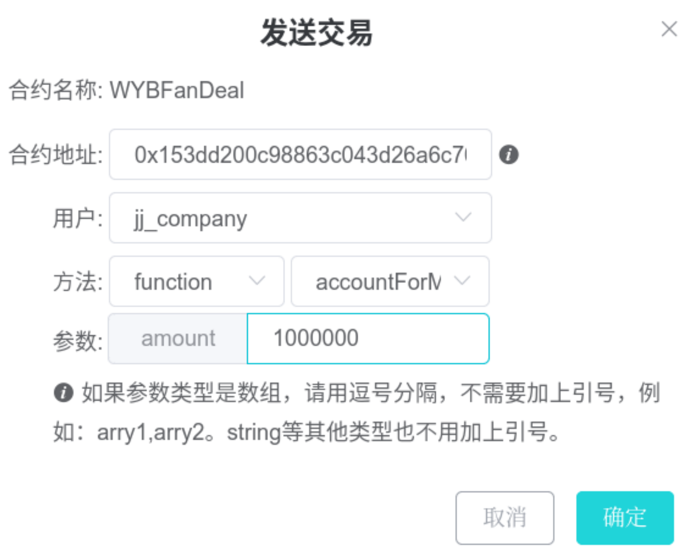

交易内容

```
status: "0x16"
from: "0x3eda1cf3d1e2e846b2052dac3ded4d413c046f31"
to: "0x153dd200c98863c043d26a6c7031d381c1b5969f"
input: "0xd54e64300000000000000000000000000000000000000000000000000000000000098
96800000000000000000000000019a652265f80acf55feef5c094feeb876a6cc762"
output: function:  payInAccount(uint256 amount,address seller)
         data:
```

```
logs: []
logsBloom: "0x0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000"
statusOK: false
blockNumberRaw: "0xe2"
transactionIndexRaw: "0x0"
gasUsedRaw: "0x602b"
}
```

3.2.6 使用债权凭证兑换资金

以 jj_company 成功使用已有债权额向银行兑换 100 万的资金为例：



合约名称: WYBFanDeal

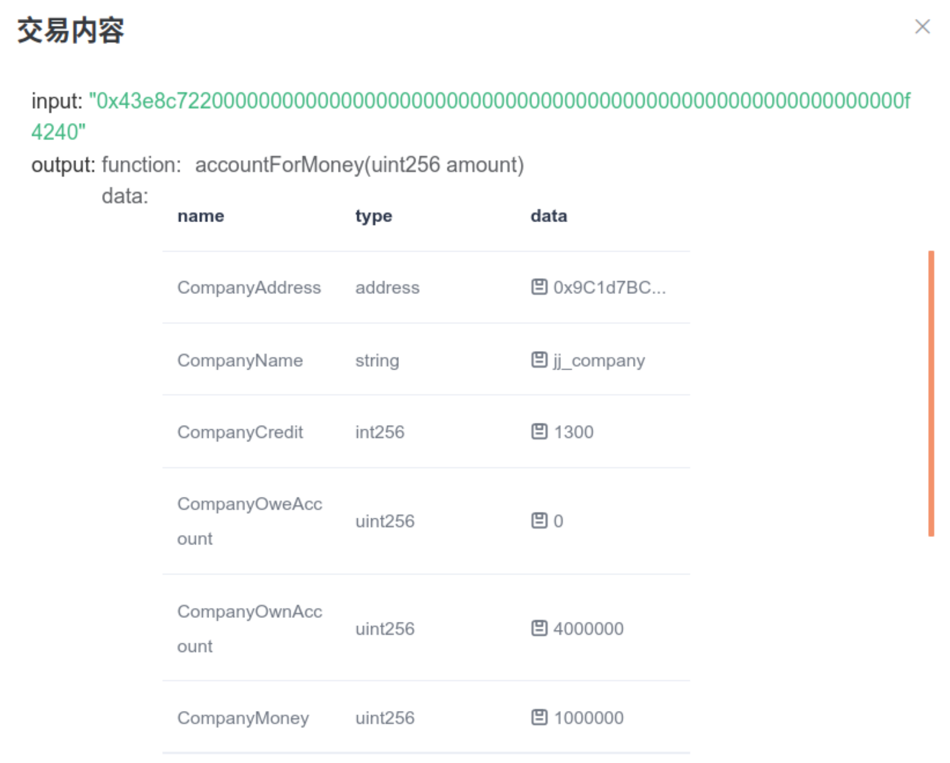
合约地址: 0x153dd200c98863c043d26a6c7

用户: jj_company

方法: function accountForM

参数: amount 1000000

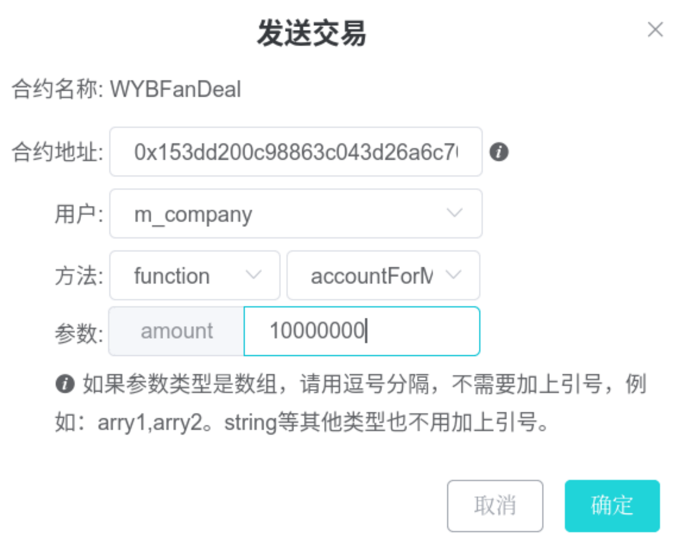
如果参数类型是数组，请用逗号分隔，不需要加上引号，例如：arry1,arry2。string等其他类型也不用加上引号。



```
input: "0x43e8c7220000000000000000000000000000000000000000000000000000000000f
4240"
output: function:  accountForMoney(uint256 amount)
        data:
```

name	type	data
CompanyAddress	address	0x9C1d7BC...
CompanyName	string	jj_company
CompanyCredit	int256	1300
CompanyOweAccount	uint256	0
CompanyOwnAccount	uint256	4000000
CompanyMoney	uint256	1000000

公司尝试兑换的资金数不能超过本司拥有的债权额，否则将导致交易失败（以 m_company 向银行兑换 1000 万的资金为例）：



合约名称: WYBFanDeal

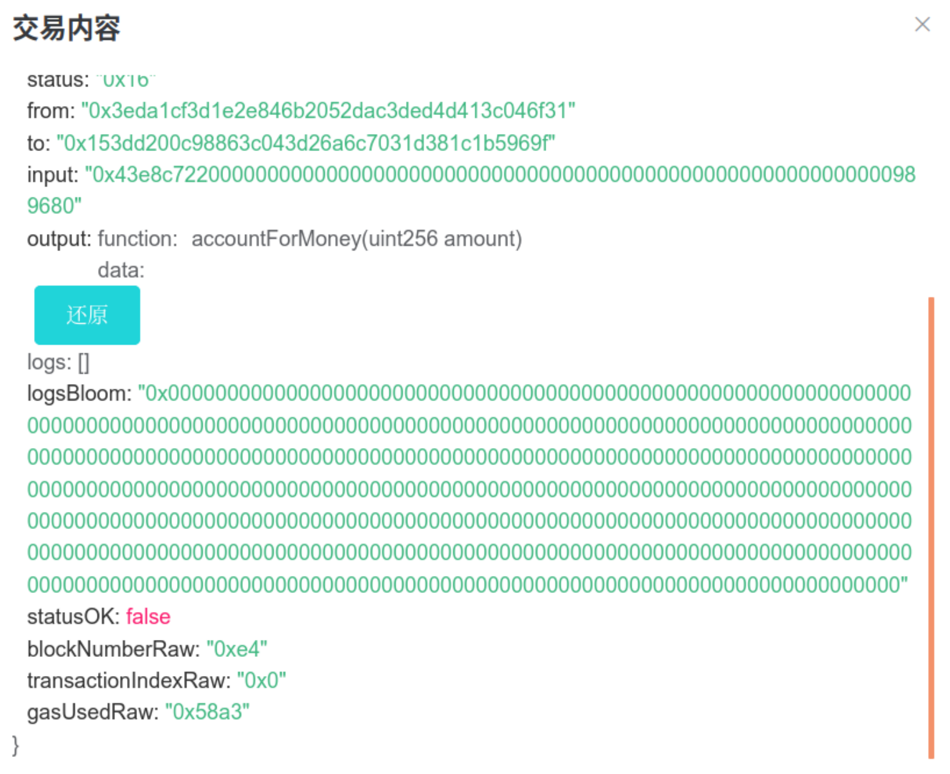
合约地址: 0x153dd200c98863c043d26a6c7

用户: m_company

方法: function accountForM

参数: amount 10000000

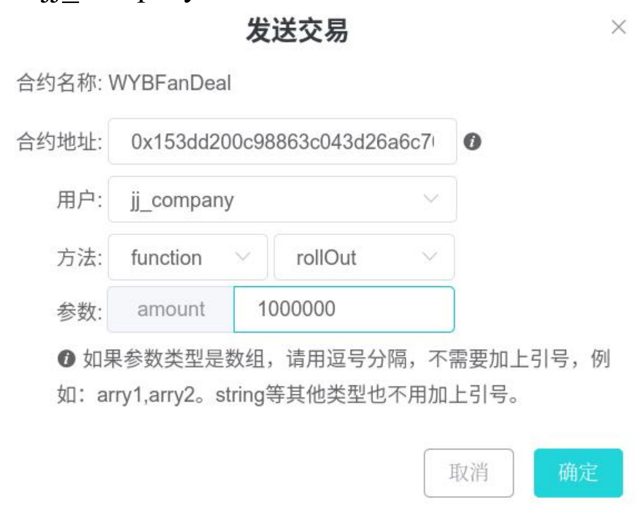
如果参数类型是数组，请用逗号分隔，不需要加上引号，例如：arry1,arry2。string等其他类型也不用加上引号。



```
status: "0x16"
from: "0x3eda1cf3d1e2e846b2052dac3ded4d413c046f31"
to: "0x153dd200c98863c043d26a6c7031d381c1b5969f"
input: "0x43e8c722000000000000000000000000000000000000000000000000000000000098
9680"
output: function:  accountForMoney(uint256 amount)
        data:
logs: []
logsBloom: "0x00000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000"
statusOK: false
blockNumberRaw: "0xe4"
transactionIndexRaw: "0x0"
gasUsedRaw: "0x58a3"
}
```

3.2.7 转出资金

以 jj_company 转出（提现）其目前拥有的全部资金（100 万）为例：



合约名称: WYBFanDeal

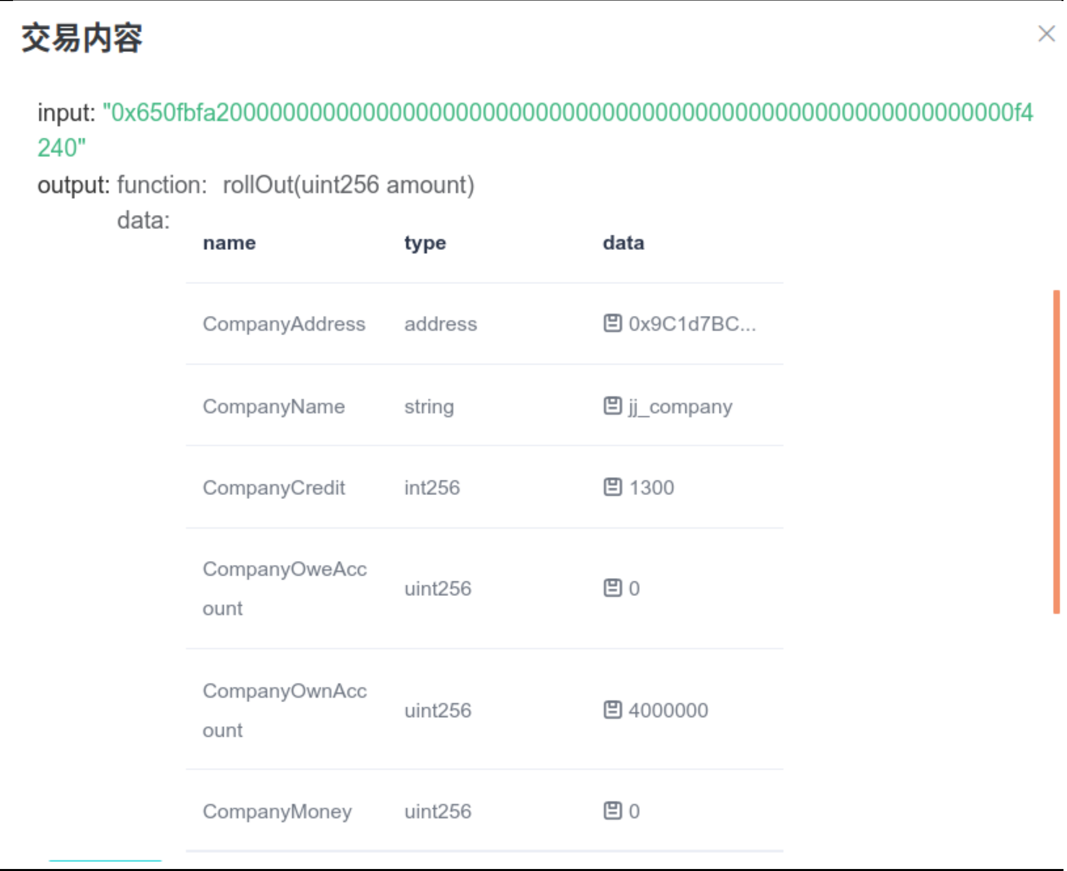
合约地址: 0x153dd200c98863c043d26a6c7

用户: jj_company

方法: function rollOut

参数: amount 1000000

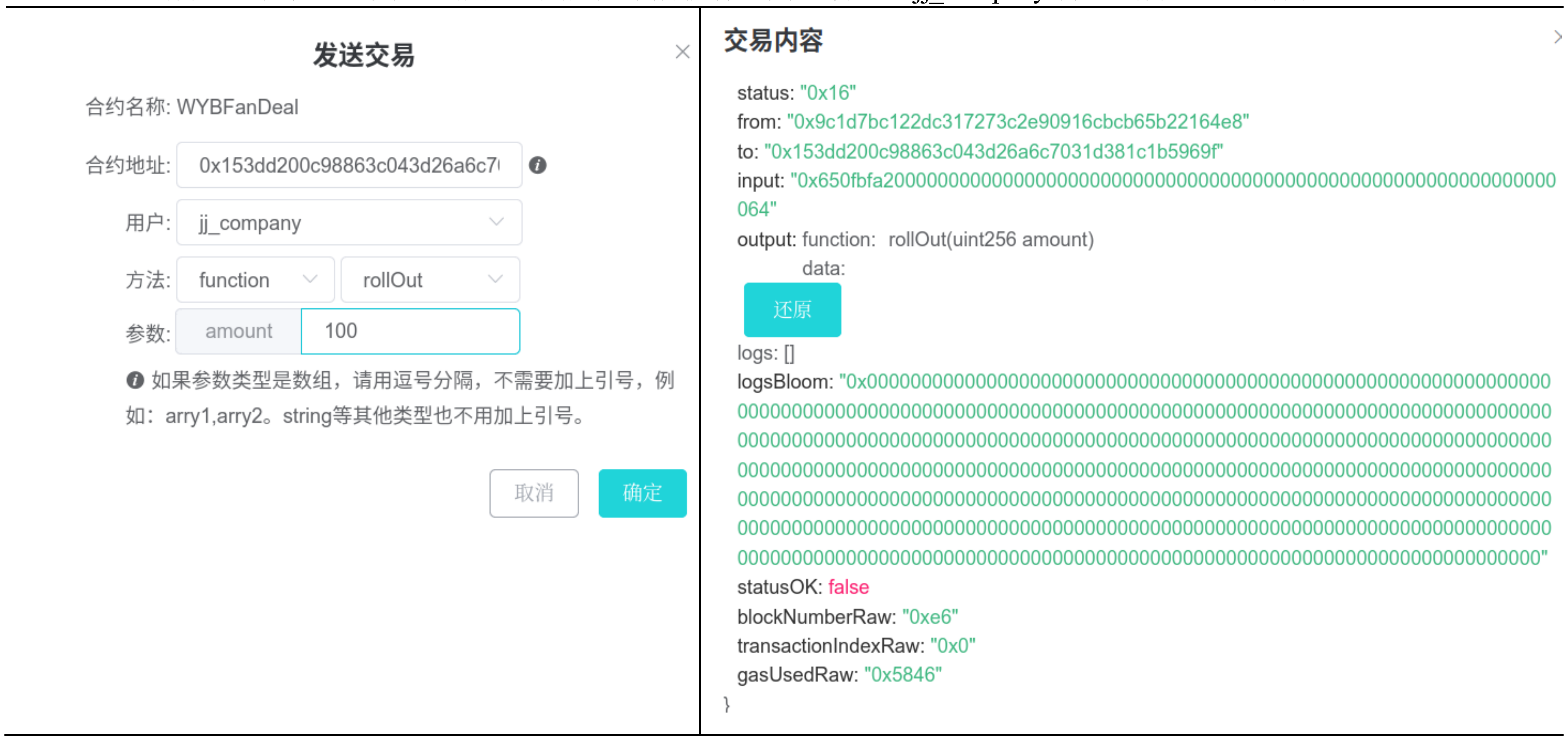
如果参数类型是数组，请用逗号分隔，不需要加上引号，例如：arry1,arry2。string等其他类型也不用加上引号。



```
input: "0x650fbfa20000000000000000000000000000000000000000000000000000000000f4
240"
output: function:  rollOut(uint256 amount)
        data:
```

name	type	data
CompanyAddress	address	0x9C1d7BC...
CompanyName	string	jj_company
CompanyCredit	int256	1300
CompanyOweAccount	uint256	0
CompanyOwnAccount	uint256	4000000
CompanyMoney	uint256	0

公司尝试转出（提现）的资金不能超过其账户中所拥有的资金额（以 jj_company 再尝试转出 100 为例）：



发送交易

合约名称: WYBFanDeal

合约地址: 0x153dd200c98863c043d26a6c7

用户: jj_company

方法: function rollOut

参数: amount 100

如果参数类型是数组，请用逗号分隔，不需要加上引号，例如：arry1,arry2。string等其他类型也不用加上引号。

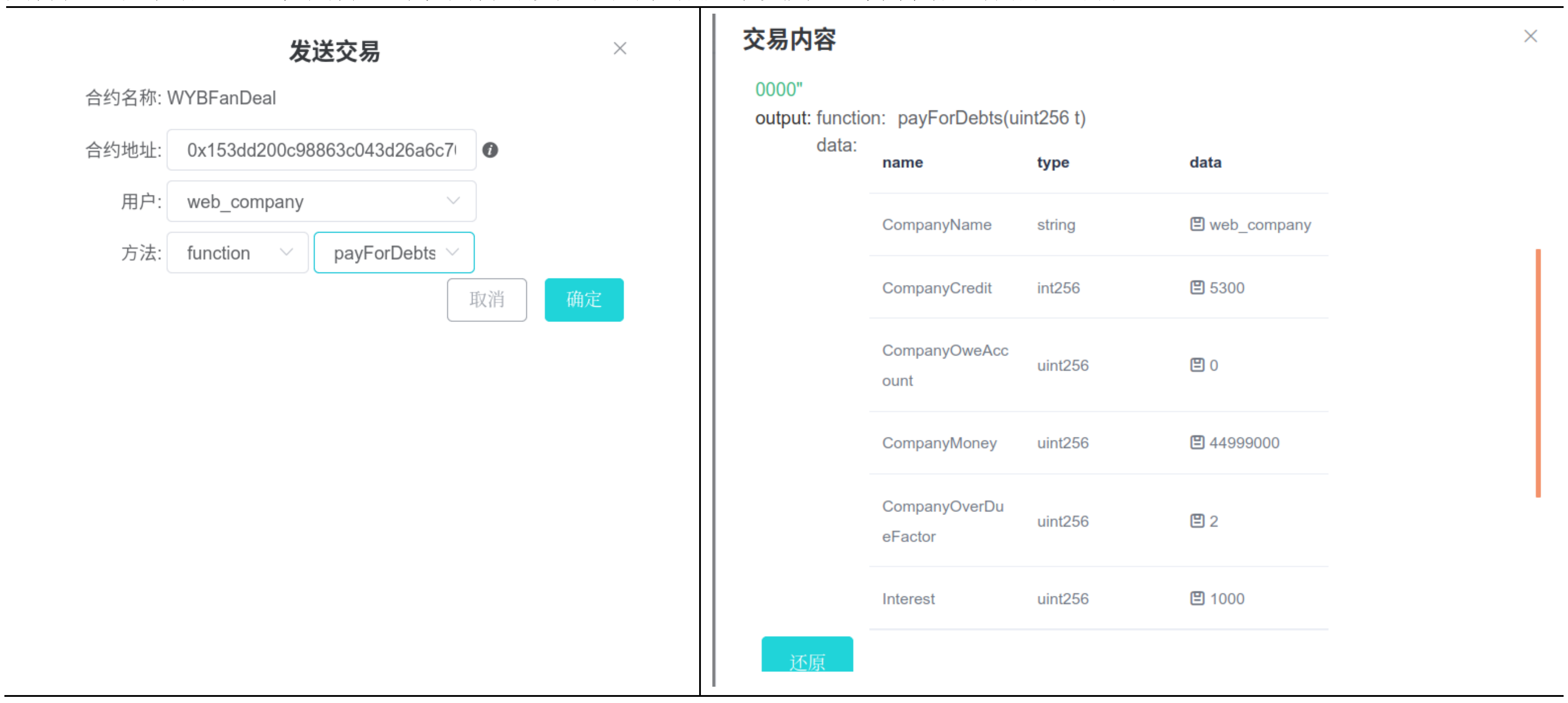
取消 确定

交易内容

```
status: "0x16"
from: "0x9c1d7bc122dc317273c2e90916cbcb65b22164e8"
to: "0x153dd200c98863c043d26a6c7031d381c1b5969f"
input: "0x650fbfa20000000000000000000000000000000000000000000000000000000000000
064"
output: function:  rollOut(uint256 amount)
         data:
还原
logs: []
logsBloom: "0x0000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000"
statusOK: false
blockNumberRaw: "0xe6"
transactionIndexRaw: "0x0"
gasUsedRaw: "0x5846"
}
```

3.2.8 兑还债权凭证

让 web_company 一次性兑付 3.2.4 中签发的债权凭证，由于目前时间已经超过 3 分钟，故而，web_company 再加上 1000 信用分后还要扣减 1000 信用分，即信用分不变，同时扣减因子变为 2，并付给银行相应的利息：



发送交易

合约名称: WYBFanDeal

合约地址: 0x153dd200c98863c043d26a6c7

用户: web_company

方法: function payForDebts

取消 确定

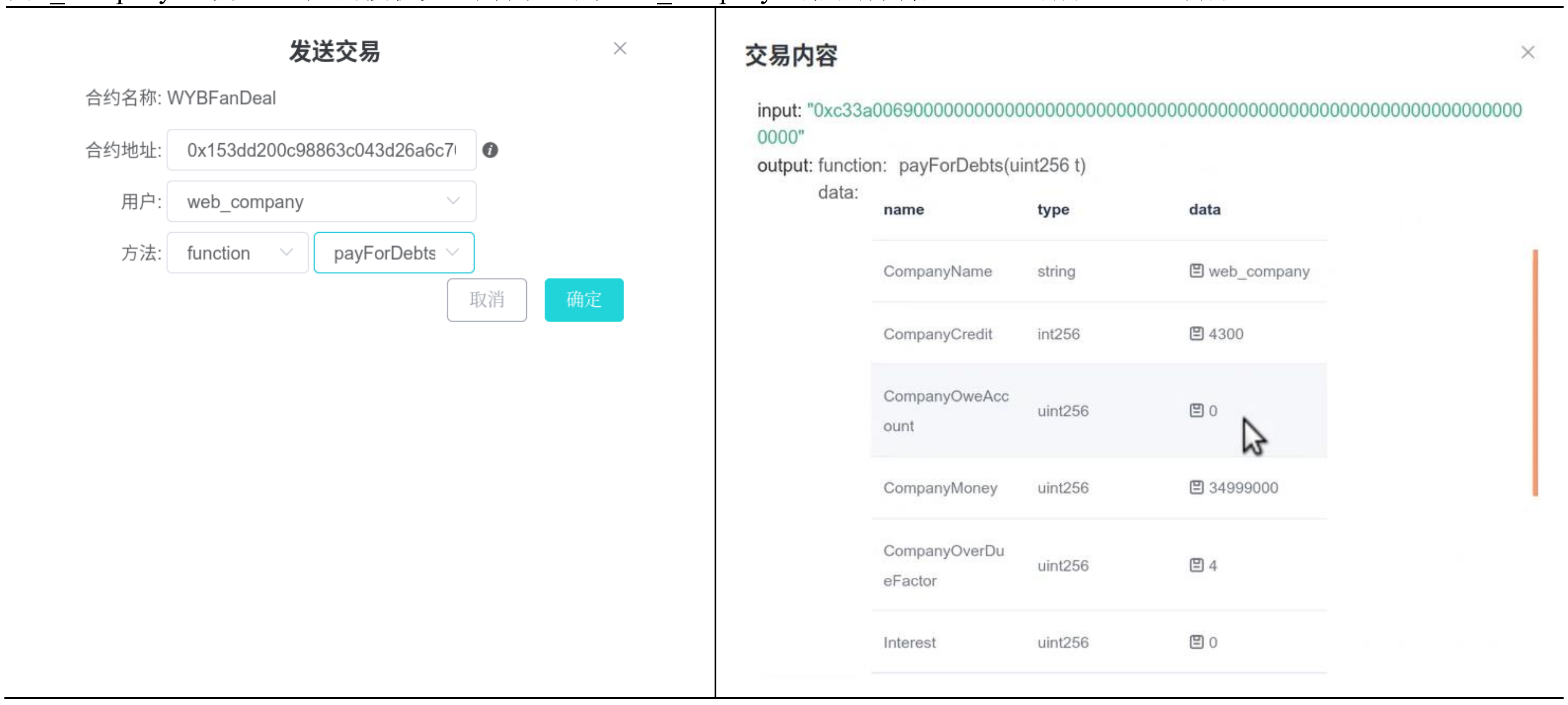
交易内容

```
0000"
output: function:  payForDebts(uint256 t)
         data:
```

name	type	data
CompanyName	string	web_company
CompanyCredit	int256	5300
CompanyOweAccount	uint256	0
CompanyMoney	uint256	44999000
CompanyOverDueFactor	uint256	2
Interest	uint256	1000

还原

再做一次实验，为了方便我们这里将还款期限改为 5 秒，让 web_company 再次超过还款期限兑还债款（仍以 web_company 向 t_company 签发 1000 万的债权凭证为例），则 web_company 的信用分将加 1000 而减 2000，即减 1000：



发送交易

合约名称: WYBFanDeal

合约地址: 0x153dd200c98863c043d26a6c7

用户: web_company

方法: function payForDebts

取消 确定

交易内容

```
input: "0xc33a00690000000000000000000000000000000000000000000000000000000000
0000"
output: function:  payForDebts(uint256 t)
         data:
```

name	type	data
CompanyName	string	web_company
CompanyCredit	int256	4300
CompanyOweAccount	uint256	0
CompanyMoney	uint256	34999000
CompanyOverDueFactor	uint256	4
Interest	uint256	0

四、界面展示

参见三 功能测试。

五、心得体会

本次实验最大的收获之一就是对供应链金融有了进一步的了解，而基于此了解基础上的**功能扩展也正是本制品的一个亮点**。在本制品中，我设计了一个信用评分机制并采用线性加分、乘性扣分的方式以提高债权凭证的可信度，同时让银行可以从签发债权凭证者处收取利息，从而鼓励银行承担风险，同时中小企业之间资金的流动又不会受大企业资金暂时的短缺影响。

此外，前期实验中也让我对 weBASE 管理平台有了进一步认识，对链的配置和部署都有了进一步掌握。

不过遗憾的是，由于个人本学期选课门数偏多，课业繁重，故未能抽出时间给本制品实现一个更高效的交互界面，也未能对底层的共识算法作进一步探讨，希望未来能有机会实现相应的功能。

六、Github 网址

https://github.com/Jie-Re/BlockChain/tree/master

七、智能合约源码

```
pragma solidity >=0.4.22 <0.7.0;

contract WYBFanDeal {
    struct Company {
        string name;
        address addr;
        int credit_rate;
        uint oweAccount;
        uint ownAccount;
        uint money;
        uint overdueFactor;
    }

    struct Voucher {
        uint256 voucherID;
        address loanTo;
        address loanFrom;
        bool isValid;
        uint amount;
        uint256 startTime;
    }

    uint256 maxVoucherID;
    uint256 minVoucherID;

    uint256 payDeadline;

    address public bank;

    mapping(address => Company) companies;
    mapping(uint256 => Voucher) vouchers;

    constructor () public {
        bank = msg.sender;
        maxVoucherID = 0;
        minVoucherID = 0;

        companies[bank].addr = bank;
        companies[bank].credit_rate = 100000;
        companies[bank].money = 1000000000;
        companies[bank].oweAccount = 0;
        companies[bank].ownAccount = 0;
    }

    function incorporateCompany(string companyName) public
    returns (address CompanyAddress, string CompanyName, int CompanyCredit,
            uint CompanyOweAccount, uint CompanyOwnAccount, uint CompanyMoney) {
        /*check first*/
        address companyAddr = msg.sender;
        require(
            companies[companyAddr].addr == 0,
            "Your company have incorporated, please don't do it again!"
        );
        /*body*/
        companies[companyAddr].addr = companyAddr;
        companies[companyAddr].name = companyName;
        companies[companyAddr].credit_rate = 800;
        companies[companyAddr].oweAccount = 0;
        companies[companyAddr].ownAccount = 0;
        companies[companyAddr].money = 0;
        companies[companyAddr].overdueFactor = 1;
        /*returns*/
        return (companies[companyAddr].addr, companies[companyAddr].name, companies[companyAddr].credit_rate,
            companies[companyAddr].oweAccount, companies[companyAddr].ownAccount, companies[companyAddr].money);
    }

    function rollIn(uint amount) public returns (address CompanyAddress, string CompanyName, int CompanyCredit,
            uint CompanyOweAccount, uint CompanyOwnAccount, uint CompanyMoney) {
        /*check first*/
        address companyAddr = msg.sender;
        require(
            companies[companyAddr].addr != 0,
            "You must incorporate your company first!"
        );
        /*body*/
        companies[companyAddr].money += amount;
        /*returns*/
        return (companies[companyAddr].addr, companies[companyAddr].name, companies[companyAddr].credit_rate,
            companies[companyAddr].oweAccount, companies[companyAddr].ownAccount, companies[companyAddr].money);
    }

    function rollOut(uint amount) public returns (address CompanyAddress, string CompanyName, int CompanyCredit,
            uint CompanyOweAccount, uint CompanyOwnAccount, uint CompanyMoney) {
        /*check first*/
        address companyAddr = msg.sender;
        require(
            companies[companyAddr].addr != 0,
            "You must incorporate your company first!"
        );
        require(
            companies[companyAddr].money >= amount,
            "You must have enough money to roll out."
        );
        /*body*/
        companies[companyAddr].money -= amount;
        /*returns*/
        return (companies[companyAddr].addr, companies[companyAddr].name, companies[companyAddr].credit_rate,
            companies[companyAddr].oweAccount, companies[companyAddr].ownAccount, companies[companyAddr].money);
    }

    function payInCash(uint amount, address seller) public returns (string CompanyName,
        int CompanyCredit, uint CompanyMoney, string SellerName, int SellerCredit, uint SellerMoney) {
        /*check first*/
        address companyAddr = msg.sender;
        require(
                companies[companyAddr].addr != 0,
                "You must incorporate your company first!"
            );
            require(
                companies[seller].addr != 0,
                "The seller must have already incorporated!"
            );
            require(
                companies[companyAddr].money >= amount,
                "You must have enough money to pay in cash."
            );
            /*body*/
            //money transfer
            companies[companyAddr].money -= amount;
            companies[seller].money += amount;
            //add credit
            companies[companyAddr].credit_rate += (int)(amount / 10000);
            companies[seller].credit_rate += (int)(amount / 10000);
            /*returns*/
            return (companies[companyAddr].name, companies[companyAddr].credit_rate, companies[companyAddr].money,
                companies[seller].name, companies[seller].credit_rate, companies[seller].money);
        }

        function issueDebtCertificate(uint amount, address seller) public returns (string CompanyName,
            uint CompanyOweAccount, int CompanyCredit, string SellerName, uint SellerOwnAccount,
            int SellerCredit) {
            /*check first*/
            address companyAddr = msg.sender;
            require(
                companies[companyAddr].addr != 0,
                "You must incorporate your company first!"
            );
            require(
                companies[seller].addr != 0,
                "The seller must have already incorporated!"
            );
            require(
                companies[companyAddr].credit_rate >= 5000,
                "You cannot issue a debt certificate."
            );
            /*body*/
            //voucher
            maxVoucherID += 1;
            vouchers[maxVoucherID].voucherID = maxVoucherID;
            vouchers[maxVoucherID].loanFrom = companyAddr;
            vouchers[maxVoucherID].loanTo = seller;
            vouchers[maxVoucherID].isValid = true;
            vouchers[maxVoucherID].amount = amount;
            vouchers[maxVoucherID].startTime = now;
            //account transfer
            companies[companyAddr].oweAccount += amount;
            companies[seller].ownAccount += amount;
            //add seller's credit
            companies[seller].credit_rate += (int)(amount / 10000);
            /*returns*/
            return (companies[companyAddr].name, companies[companyAddr].oweAccount, companies[companyAddr].credit_rate,
                companies[seller].name, companies[seller].ownAccount, companies[seller].credit_rate);
        }

        function payInAccount(uint amount, address seller) public returns (string CompanyName,
            uint CompanyOwnAccount, string SellerName, uint SellerOwnAccount,
            int SellerCredit) {
            /*check first*/
            require(
            companies[msg.sender].addr != 0,
            "You must incorporate your company first!"
    );
    require(
            companies[seller].addr != 0,
            "The seller must have already incorporated!"
    );
    require(
            companies[msg.sender].ownAccount >= amount,
            "You must own enough debt certificate to pay."
    );
    /*body*/
    //get enough vouchers whose loanTo is msg.sender and isValid is true
    uint tempAccount = 0;
    uint count = 0;
    for (uint256 i = minVoucherID + 1; i <= maxVoucherID; i++) {
            if (vouchers[i].loanTo == msg.sender && vouchers[i].isValid == true) {
                    //temporary counters
                    tempAccount += vouchers[i].amount;
                    //old voucher processing
                    vouchers[i].isValid = false;
                    //add vouchers
                    count += 1;
                    vouchers[maxVoucherID + count].voucherID = maxVoucherID + count;
                    vouchers[maxVoucherID + count].isValid = true;
                    vouchers[maxVoucherID + count].startTime = vouchers[i].startTime;
                    //transfer the debt certificate
                    vouchers[maxVoucherID + count].loanTo = seller;
                    vouchers[maxVoucherID + count].loanFrom = vouchers[i].loanFrom;
                    if (tempAccount > amount) {
                            vouchers[maxVoucherID + count].amount = vouchers[i].amount - (tempAccount - amount);
                            //new another voucher
                            count += 1;
                            vouchers[maxVoucherID + count].voucherID = maxVoucherID + count;
                            vouchers[maxVoucherID + count].loanTo = msg.sender;
                            vouchers[maxVoucherID + count].loanFrom = vouchers[i].loanFrom;
                            vouchers[maxVoucherID + count].isValid = true;
                            vouchers[maxVoucherID + count].amount = tempAccount - amount;
                            vouchers[maxVoucherID + count].startTime = vouchers[i].startTime;
                            break;
                    }
                    else if (tempAccount == amount) {
                            vouchers[maxVoucherID + count].amount = vouchers[i].amount;
                            break;
                    }
            }
    }
    //update maxVoucherID
    maxVoucherID += count;
    //update minVoucherID
    for (i = minVoucherID + 1; i <= maxVoucherID; i ++) {
            if (vouchers[i].isValid == true) {
                    break;
            }
            minVoucherID += 1;
    }
    //account transfer
    companies[msg.sender].ownAccount -= amount;
    companies[seller].ownAccount += amount;
    //add seller's credit
    companies[seller].credit_rate += (int)(amount / 10000);
    /*returns*/
    return (companies[msg.sender].name, companies[msg.sender].ownAccount,
            companies[seller].name, companies[seller].ownAccount, companies[seller].credit_rate);
```

```
}

function accountForMoney(uint amount) public returns (address CompanyAddress, string CompanyName,
int CompanyCredit, uint CompanyOweAccount, uint CompanyOwnAccount, uint CompanyMoney) {
    /*check first*/
    address companyAddr = msg.sender;
    require(
        companies[companyAddr].addr != 0,
        "You must incorporate your company first!"
    );
    require(
        companies[companyAddr].ownAccount >= amount,
        "You must have enough account to exchange for money"
    );
    /*body*/
    //get enough vouchers whose loanTo is companyAddr and isValid is true
    uint tempAccount = 0;
    uint count = 0;
    for (uint256 i = minVoucherID + 1; i <= maxVoucherID; i++)  {
        if (vouchers[i].loanTo == companyAddr && vouchers[i].isValid == true) {
            //temporary counters
            tempAccount += vouchers[i].amount;
            //old voucher processing
            vouchers[i].isValid = false;
            //add vouchers
            count += 1;
            vouchers[maxVoucherID + count].voucherID = maxVoucherID + count;
            vouchers[maxVoucherID + count].isValid = true;
            vouchers[maxVoucherID + count].startTime = vouchers[i].startTime;
            //transfer the debt certificate to bank
            vouchers[maxVoucherID + count].loanTo = bank;
            vouchers[maxVoucherID + count].loanFrom = vouchers[i].loanFrom;
            if (tempAccount > amount) {
                vouchers[maxVoucherID + count].amount = vouchers[i].amount - (tempAccount - amount);
                count += 1;
                vouchers[maxVoucherID + count].voucherID = maxVoucherID + count;
                vouchers[maxVoucherID + count].loanTo = companyAddr;
                vouchers[maxVoucherID + count].loanFrom = vouchers[i].loanFrom;
                vouchers[maxVoucherID + count].isValid = true;
                vouchers[maxVoucherID + count].amount = tempAccount - amount;
                vouchers[maxVoucherID + count].startTime = vouchers[i].startTime;
                break;
            }
            else if (tempAccount == amount) {
                vouchers[maxVoucherID + count].amount = vouchers[i].amount;
                break;
            }
        }
    }
    //update maxVoucherID
    maxVoucherID += count;
    //update minVoucherID
    for (i = minVoucherID + 1; i <= maxVoucherID; i ++)  {
        if (vouchers[i].isValid == true) {
            break;
        }
        minVoucherID += 1;
    }
    //account transfer
    companies[companyAddr].ownAccount -= amount;
    companies[bank].ownAccount += amount;
    //add the company's money while subtract bank's money
    companies[companyAddr].money += amount;
    companies[bank].money -= amount;
```

```
        /*returns*/
        return (companies[companyAddr].addr, companies[companyAddr].name, companies[companyAddr].credit_rate,
                companies[companyAddr].oweAccount, companies[companyAddr].ownAccount, companies[companyAddr].money);
}

function payForDebts(uint t) public returns (string CompanyName, int CompanyCredit, uint CompanyOweAccount,
uint CompanyMoney, uint CompanyOverDueFactor, uint Interest) {
        /*check first*/
        address companyAddr = msg.sender;
        require(
                companies[companyAddr].addr != 0,
                "You must incorporate your company first!"
        );
        //compute the interest to bank
        uint maxInterest = companies[companyAddr].oweAccount * 30 / 10000;
        require(
                companies[companyAddr].money >= companies[companyAddr].oweAccount + maxInterest,
                "You must have enough money to pay your debts and interests off!"
        );
        /*body*/
        //store company's oweAccount temporarily
        t = companies[companyAddr].oweAccount;
        //add company's credit
        companies[companyAddr].credit_rate += (int)(companies[companyAddr].oweAccount / 10000);
        bool isOverdue = false;
        uint totalInterest = 0;
        //pay off
        for (uint256 i = minVoucherID + 1; i <= maxVoucherID; i ++) {
                if (vouchers[i].isValid == true && vouchers[i].loanFrom == companyAddr) {
                        vouchers[i].isValid = false;
                        companies[companyAddr].money -= vouchers[i].amount;
                        companies[companyAddr].oweAccount -= vouchers[i].amount;
                        companies[vouchers[i].loanTo].money += vouchers[i].amount;
                        companies[vouchers[i].loanTo].ownAccount -= vouchers[i].amount;
                        //the interest to bank
                        uint daysAfter = (now - vouchers[i].startTime) / (24 * 3600);
                        uint interest = vouchers[i].amount * daysAfter / 10000;
                        companies[companyAddr].money -= interest;
                        companies[bank].money += interest;
                        totalInterest += interest;
                        //check isOverdue
                        if (now - vouchers[i].startTime > payDeadline) {
                                isOverdue = true;
                        }
                }
        }
        //if overdue subtract company's credit
        if (isOverdue == true) {
                companies[companyAddr].credit_rate -= (int)(t / 10000 * companies[companyAddr].overdueFactor);
                companies[companyAddr].overdueFactor *= 2;
        }
        //update minVoucherID
        for (i = minVoucherID + 1; i <= maxVoucherID; i ++) {
                if (vouchers[i].isValid == true) {
                        break;
                }
                minVoucherID += 1;
        }
        /*returns*/
        return (companies[companyAddr].name, companies[companyAddr].credit_rate, companies[companyAddr].oweAccount,
        companies[companyAddr].money, companies[companyAddr].overdueFactor, totalInterest);
}

function setDeadline(uint deadline) public returns (uint DeadlineInSeconds){
        require(
            msg.sender == bank,
            "Only bank can call this function"
        );
        require(
            deadline < 30 * 1 days,
            "The max deadline is 30 days"
        );
        payDeadline = deadline;
        return payDeadline;
    }
}
```