

# 没有中奸商赚差价——商品拍卖业务网络

这是一个交互式、分布式的商品拍卖的解决方案。本解决方案利用了区块链技术，利用了区块链的去中心化特点，提供了所有事务的共享，公开、不可变和透明的拍卖历史记录，以构建具有信任、责任和透明度的拍卖系统。方便监管机构对拍卖数据进行追溯、监督，防止有人抵赖或者中间商从拍卖会中恶意赚取差价。在这里，拍卖商可以举办拍卖活动；客户可以通过拍卖活动拍下商品，也可以对自己的帐户进行余额充值。

## 业务网络模型定义

### 参与者(Participants)

- `Customer`：客户（或顾客），也就是拍卖的参与者，可以是买方或者卖方，标识符为客户的邮箱
- `Auctioneer`：拍卖商，相当于一次拍卖的主持人，标识符也是邮箱

### 资产(Assets)

- `Product`：可以被拍卖的物品、商品，标识符为其名称
- `AuctionList`：一次拍卖的清单，标识符为这次拍卖的ID

### 事务(Transactions)

- `MakeAuctionList`：添加新拍卖
- `AuctionOffer`：客户发出竞拍物品请求
- `CloseBidding`：结束一次拍卖
- `Recharge`：客户帐户充值

### 事件(Events)

- `AuctionFinishNotification`：结束拍卖的事件，在事件 `CloseBidding` 中发出
- `OfferPostedNotification`：成功发出竞拍请求的事件，在事件 `AuctionOffer` 中发出
- `RechargeNotification`：充值成功的事件，在事件 `Recharge` 中发出
- `NewAuctionNotification`：添加新拍卖成功的事件，在事件 `MakeAuctionList` 中发出

## 业务网络功能介绍

在这个虚拟的拍卖场景中，假定参与者 `Customer` 和 `Auctioneer` 都用邮箱来区分，每个帐户都有自己的昵称和真实姓名，昵称的设置是防止真实姓名泄露而带来的隐私泄露。`Customer` 是客户，在一次拍卖中充当交易的参与者，每位客户的帐户都有自己的余额；`Auctioneer` 是拍卖商，充当一次拍卖的主持人（或者赞助商），不从中赚取任何差价。

在这里，资产分为 `Product` 和 `AuctionList`。拍卖中交易的对象是 `Product`，`Product` 是商品，以它的名称作为区分。`Customer` 可以随时登记自己的商品信息，商品信息包括商品名称和所有者（登记这件商品的人，也就是这位 `Customer`）。`AuctionList` 是拍卖会上一次拍卖的清单，一次拍卖只对应一个商品（一个商品可以对应很多次拍卖）。包括的信息有：这次拍卖的ID、这次拍卖所对应的商品、本次拍卖的描述（可选）、拍卖商、本次拍卖的状态、不同的竞拍请求（可以没有，也可以有多个，同一个人可以对同一次拍卖发出多个请求）、本次拍卖开始的时间。本次拍卖的状态有三种：待售（本次拍卖正在进行，状态编号 `FOR_SALE`）、成功售出（状态编号 `SOLD`）、非正常结束（状态编号 `ENDED`，表明在结束时此次拍卖没有 `Customer` 发起竞拍请求或者出价最高者出的价格未达到底价）。

事务分为 `MakeAuctionList`、`AuctionOffer`、`CloseBidding`、`Recharge`。`MakeAuctionList` 是通过输入底价、拍卖描述（可选）、商品名称、拍卖商来创建新的拍卖并自动生成新拍卖的ID（格式为yyyyymmdd\_当日拍卖次数，例：`20190718_2`，表明是在北京时间2019年7月18日在拍卖会上发起的第二次拍卖），状态编号自动置为 `FOR_SALE`。`AuctionOffer` 是对某一场次的拍卖（必须是未结束的拍卖）发出竞价请求，输入竞拍价格（必须大于0且不超过竞拍人的余额）、竞拍场次ID和竞拍人来发出请求。`CloseBidding` 是结束某一场次的拍卖（必须是未结束的拍卖），通过输入这个拍卖的ID；如果出价最高的竞拍人出的价格达到底价，则本次拍卖所对应的商品所有权转移至这个竞拍人，竞拍人的余额被扣除他所出的价格，商品所对应的原来的所有者的余额加上这个竞拍人所出的价格。`Recharge` 是对某一个 `Customer` 帐户进行余额充值，充值的余额必须大于0；通过输入帐户所对应的邮箱和充值的余额进行充值。对一个 `Customer` 来说，增加帐户余额有两种方式：要么拍卖商品，要么充值余额。

事件的功能同上节。

所有修改后的资产或参与者等都在各自的注册表中更新。

详细的模型定义在文件 `auction.cto` 中，事务逻辑定义在 `logic.js` 中，事务逻辑所用到的查询在 `queries.qry` 中，访问权限控制在 `permissions.pal` 中。

## 测试业务网络模型

测试此业务网络定义（只演示操作正确的情况）：

在 `Auctioneer` 的注册表中，创建一个新实例。

```
1 {
2   "$class": "org.product.auction.Auctioneer",
3   "email": "1234@1234.com",
4   "customerName": {
5     "$class": "org.product.auction.CustomerName",
6     "nickName": "Bill",
7     "realName": "Bill Gates"
8   }
9 }
```

在 `Customer` 的注册表中，创建两个实例。

```
1 {
2   "$class": "org.product.auction.Customer",
3   "balance": 5000,
4   "email": "4567@4567.com",
5   "customerName": {
6     "$class": "org.product.auction.CustomerName",
7     "nickName": "Mikeeee",
8     "realName": "Mike"
9   }
10 }
11 {
12   "$class": "org.product.auction.Customer",
13   "balance": 5000,
14   "email": "7890@7890.com",
15   "customerName": {
16     "$class": "org.product.auction.CustomerName",
17     "nickName": "Nikeabcde",
```

```
18     "realName": "Nike"
19   }
20 }
```

在 `Product` 的注册表中，创建 `4567@4567.com` 的新商品。

```
1 {
2   "$class": "org.product.auction.Product",
3   "productName": "product",
4   "owner": "resource:org.product.auction.Customer#4567@4567.com"
5 }
```

通过事件 `MakeAuctionList`，为名称为 `product` 的商品创建新的由拍卖商 `1234@1234.com` 主持的新拍卖，底价 `3500`。

```
1 {
2   "$class": "org.product.auction.MakeAuctionList",
3   "reservePrice": 3500,
4   "product": "resource:org.product.auction.Product#product",
5   "auctioneer": "resource:org.product.auction.Auctioneer#1234@1234.com"
6 }
```

一旦新的拍卖 `AuctionList` 被创建（并且处于 `FOR_SALE` 状态），`Customer` 就可以提交 `AuctionOffer` 事件以参与这次拍卖的竞价。假设生成的拍卖ID为 `20190719_2`。

提交 `AuctionOffer` 事务以发出竞拍请求。

```
1 {
2   "$class": "org.product.auction.AuctionOffer",
3   "bidPrice": 2000,
4   "auctionList": "org.product.auction.AuctionList#20190719_2",
5   "member": "resource:org.product.auction.Customer#4567@4567.com"
6 }
7 {
8   "$class": "org.product.auction.AuctionOffer",
9   "bidPrice": 3500,
10  "auctionList": "org.product.auction.AuctionList#20190719_2",
11  "member": "resource:org.product.auction.Customer#7890@7890.com"
12 }
```

要结束这次拍卖，提交 `CloseBidding` 事务：

```
1 {
2   "$class": "org.product.auction.CloseBidding",
3   "auctionList": "resource:org.product.auction.AuctionList#20190719_2"
4 }
```

要查看商品是否已售出，即这次拍卖是否已经成功结束，则可以通过 `Product` 的注册表检查商品所有者是否变化。在前面所有竞价请求中，最高竞拍价格大于等于底价，因此商品成功售出。在 `Product` 注册表中会发现所有者邮箱从 `4567@4567.com` 变为 `7890@7890.com`。

通过 Customer 的注册表检查每个帐户的余额。可以看到买方的余额 7890@7890.com 已经扣除 3500，而卖方的余额 4567@4567.com 已增加了 3500。

此次拍卖状态由 FOR\_SALE 变为 SOLD。

要为帐户 7890@7890.com 充值，则提交 Recharge 事务：

```
1 {
2   "$class": "org.product.auction.Recharge",
3   "amount": 1000,
4   "customer": "resource:org.product.auction.Customer#7890@7890.com"
5 }
```

然后这位 Customer 的余额会增加1000。

## 前端页面

- 拍卖会页面

⚙ 没有中奸商赚差价

🏠 拍卖

💰 余额充值

👤 客户

📦 拍卖的商品

👥 拍卖商

## 拍卖会

享受拍卖的乐趣

Step 1: 添加新拍卖

底价

大于0的任意金额即可

拍卖交易的最低金额

拍卖描述 (可选)

可选，不限字数

可描述要拍卖的物品特征等

要拍卖的物品

请输入物品名

Step 2: 参加一个竞拍

拍卖场次ID

请输入正确的拍卖场次ID

ID格式为yyyymmdd\_本日场次

竞拍价格

必须大于0且小于个人帐户余额

要购买物品的价格

请输入你的邮箱

例: example@example.com

输入正确格式的邮箱

提交竞拍请求清除信息

- 余额管理充值页面

没有中奸商赚差价

拍卖

余额管理

客户

拍卖的商品

拍卖商

余额管理

在此管理客户余额

在线充值

邮箱

@ 请输入您的合法邮箱

充值金额

\$ 请输入充值金额

确认充值 重新输入

充值记录

序号	邮箱	充值金额
1	111	20000
2	email@email.com	10000
3	abc@abc.com	1000

客户页面

没有中奸商赚差价

拍卖

余额充值

客户

拍卖的商品

拍卖商

客户

在此输入信息

基本内容

balance

\$ 在此输入余额 .00

email

@ 在此输入邮箱

nickName

在此输入昵称

realName

在此输入真实名称

提交信息 重置信息

客户信息表

#	balance	email	nickName	rea
---	---------	-------	----------	-----

拍卖的商品页面

⚙ 没有中奸商赚差价

拍卖

余额充值

客户

拍卖的商品

拍卖商

拍卖的商品 ——商品列表

商品名称

例:传国玉玺

所有者邮箱

例:abc@abc.com

Submit Button

商品列表

商品名称	所有者邮箱
product	abc@abc.com
product1	email@email.com
product2	abc@abc.com
product3	email@email.com
product4	qwerty@aaa.com
product5	qwerty@aaa.com

- 拍卖商页面

⚙ 没有中奸商赚差价

拍卖

余额充值

客户

拍卖的商品

拍卖商

拍卖商 Best form elements.

添加拍卖商

Nick Name

Real Name

Email

提交

拍卖商列表

Nick Name	Real Name
ooooooo	哦哦哦
travis	scott
adasd	asfas
string1	string1
el	ablimit
new	new
paimaishang	拍卖商
asfasf	afas
bill	比尔盖茨
a	212
ablimi	afsaf
500	-500

## 技术栈

- 后端：HyperLedger Composer/Fabric
- 前端：jQuery、Bootstrap
- 数据交互示意图：

