分析

受众: 架构师，应用和合约开发者，业务专家

让我们更详细的分析商业票据。MagnetoCorp 和 DigiBank 等 PaperNet 参与者使用商业票据交易来实现其业务目标 - 让我们检查商业票据的结构以及随着时间推移影响票据结构的交易。我们还将根据网络中组织之间的信任关系，考虑 PaperNet 中的哪些组织需要签署交易。稍后我们将关注买家和卖家之间的资金流动情况; 现在，让我们关注 MagnetoCorp 发行的第一个票据。

商业票据生命周期

票据 00001 是 5 月 31 号由 MagnetoCorp 发行的。花点时间来看看该票据的第一个**状态**，它具有不同的属性和值：

```
Issuer = MagnetoCorp
Paper = 00001
Owner = MagnetoCorp
Issue date = 31 May 2020
Maturity = 30 November 2020
Face value = 5M USD
Current state = issued
```

该票据的状态是 **发行** 交易的结果，它使得 MagnetoCorp 公司的第一张商业票据面世！注意该票据在今年晚些时候如何兑换面值 500 万美元。当票据 00001 发行后 `Issuer` 和 `Owner` 具有相同的值。该票据有唯一标识 `MagnetoCorp00001` ——它是 `Issuer` 属性和 `Paper` 属性的组合。最后，属性 `Current state = issued` 快速识别了 MagnetoCorp 票据 00001 在它生命周期中的阶段。

发行后不久，该票据被 DigiBank 购买。花点时间来看看由于**购买**交易，同一个商业票据如何发生变化：

```
Issuer = MagnetoCorp
Paper = 00001
Owner = DigiBank
Issue date = 31 May 2020
Maturity date = 30 November 2020
Face value = 5M USD
Current state = trading
```

最重要的变化是 `Owner` 的改变——票据初始拥有者是 `MagnetoCorp` 而现在是 `DigiBank`。我们可以想象该票据后来如何被出售给 BrokerHouse 或 HedgeMatic，以及相应的变更为相应的 `Owner`。注意 `Current state` 允许我们轻松的识别该票据目前状态是 `trading`。

6 个月后，如果 DigiBank 仍然持有商业票据，它就可以从 MagnetoCorp 那里兑换：

```
Issuer = MagnetoCorp
Paper = 00001
Owner = MagnetoCorp
Issue date = 31 May 2020
Maturity date = 30 November 2020
```

```
Face value = 5M USD
Current state = redeemed
```

最终的**兑换**交易结束了这个商业票据的生命周期——它可以被认为票据已经终止。通常必须保留已兑换的商业票据的记录，并且 `redeemed` 状态允许我们快速识别这些。通过将 `Owner` 跟交易创建者的身份进行比较，一个票据的 `Owner` 值可以被用来在**兑换**交易上进行访问控制。Fabric 通过 `getCreator()` chaincode API来对此提供支持。如果使用 Go 语言的链码， 客户端身份链码库来获取交易创建者额外的属性。

交易

我们已经看到票据 00001 的生命周期相对简单——由于**发行**，**购买**和**兑换**交易，它在 `issued`，`trading` 和 `redeemed` 状态之间转移。

这三笔交易由 MagnetoCorp 和 DigiBank（两次）发起，并推动了 00001 票据的状态变化。让我们更详细地看一下影响票据的交易：

发行

检查 MagnetoCorp 发起的第一笔交易：

```
Txn = issue
Issuer = MagnetoCorp
Paper = 00001
Issue time = 31 May 2020 09:00:00 EST
Maturity date = 30 November 2020
Face value = 5M USD
```

观察**发行**交易是如何具有属性和值的结构的。这个交易的结构跟票据 00001 的结构是不同的但是非常匹配的。那是因为他们是不同的事情——票据 00001 反应了 PaperNet state，是作为**发行**交易的结果。这是在**发行**交易背后的逻辑（我们是看不到的），它带着这些属性并且创建了票据。因为交易**创建**了票据，着意味着在这些结构之间具有着非常密切的关系。

在这个**发行**的交易中唯一被引入的组织是 MagnetoCorp。很自然的，MagnetoCorp 需要对交易进行签名。通常，一个票据的发行者会被要求在发行一个新的票据的交易上提供批准。

购买

接下来，检查**购买**交易，将票据 00001 的所有权从 MagnetoCorp 转移到 DigiBank：

```
Txn = buy
Issuer = MagnetoCorp
Paper = 00001
Current owner = MagnetoCorp
New owner = DigiBank
Purchase time = 31 May 2020 10:00:00 EST
Price = 4.94M USD
```

看一下**购买**交易如何减少票据中最终的属性。因为交易只能**修改**该票据。只有 `New owner = DigiBank` 改变了；其他所有的都是相同的。这没关系——关于**购买**交易最重要的是所有权的变更，事实上，在这次交易中，有一份对该票据当前所有者 MagnetoCorp 的认可。

你可能会奇怪为什么 `Purchase time` 和 `Price` 属性没有在票据 00001 中体现？ 这要回到交易和票据之间的差异。494 万美元的价格标签实际上是交易的属性，而不是票据的属性。花点时间来思考一下这两者的不同；它并不像它看上去的那么明显。稍后我们会看到账本会记录这些信息片段——影响票据的所有交易历史，和它最新的状态。清楚这些信息分离非常重要。

同样值得注意的是票据 00001 可能会被买卖多次。尽管在我们的场景中略微跳过了一点，我们来检查一下如果票据 00001 变更了所有者我们**可能**会看到什么。

如果 BigFund 购买：

```
Txn = buy
Issuer = MagnetoCorp
Paper = 00001
Current owner = DigiBank
New owner = BigFund
Purchase time = 2 June 2020 12:20:00 EST
Price = 4.93M USD
```

接着由 HedgeMatic 购买：

```
Txn = buy
Issuer = MagnetoCorp
Paper = 00001
Current owner = BigFund
New owner = HedgeMatic
Purchase time = 3 June 2020 15:59:00 EST
Price = 4.90M USD
```

看看票据所有者如何变化，以及在我们的例子中，价格如何变化。你能想到 MagnetoCorp 商业票据价格会降低的原因吗?

很显然，**购买**交易要求售卖组织和购买组织都要对该交易进行离线签名，这是证明参与交易的双方共同同意该交易的证据。

兑换

票据 00001 **兑换**交易代表了它生命周期的结束。在我们相对简单的例子中，HedgeMatic 启动将商业票据转回 MagnetoCorp 的交易：

```
Txn = redeem
Issuer = MagnetoCorp
Paper = 00001
Current owner = HedgeMatic
Redeem time = 30 Nov 2020 12:00:00 EST
```

再次注意**兑换**交易有更少的属性；票据 00001 所有更改都可以通过兑换交易逻辑计算数据：`Issuer` 将成为新的所有者，`Current state` 将变成 `redeemed`。在我们的例子中指定了 `Current owner` 属性，以

便可以针对当前的票据持有者进行检查。

从信任的角度来说，跟**购买**交易一样，也可以应用到**兑换**交易中：参与交易的双方组织都需要对它离线签名。

账本

在本主题中，我们已经看到交易和产生的票据状态是 PaperNet 中两个重要的概念。的确，我们将会在任何一个 Hyperledger Fabric 分布式账本中看到这些基本元素——包含了当前所有对象最新状态的世界状态和记录了所有交易历史并能归集出最新世界状态的区块链。

在交易上必须的批准通过规则被强制执行，这个在一个交易被附加到账本之前被评估。只有当需要的签名被展示的时候，Fabric 才会接受一个交易作为有效的交易。

你现在处在一个很棒的地方，将这些想法转化为智能合约。如果您的编程有点生疏，请不要担心，我们将提供了解程序代码的提示和指示。掌握商业票据智能合约是设计自己的应用程序的第一个重要步骤。或者，你是一个有一点编程经验的业务分析师，不要害怕继续深入挖掘！