Enhancing the trade lifecycle with blockchain

Using Hyperledger Fabric blockchain to streamline and modernize the trade order-to-settlement process

Lauren Stephanian

July 18, 2017

This article shows how a bank or other large trading institution can use a Hyperledger Fabric blockchain at each stage in the trade lifecycle -- from order to settlement -- to streamline and enhance the process for all participants.

This article shows how a bank or other large trading institution can use a Hyperledger Fabric blockchain at each stage in the trade lifecycle — from order to settlement — to streamline and enhance the process for all participants. You'll see, via practical examples, how a bank could easily develop and customize a complete and seamless proprietary trading system based on Hyperledger Fabric. This will include a Client Relationship Management (CRM) system to fit the needs of sales traders and clients, a trade capture system, as well as other downstream back-office systems for exchange and settlement.

The challenge facing financial institutions

The financial industry, as a whole, has a reputation for embracing the latest and best technologies. And yet many firms in the industry still cling to legacy systems commissioned decades ago. There are two primary contributing factors.

91% of banks will have invested in blockchain solutions by 2018

Learn how blockchain solutions are rewiring financial markets. And watch this video to learn more about how IBM Blockchain can be used to streamline trade finance:

To view this video, Streamlining Trade Finance with IBM Blockchain, please access the online version of the article. If this article is in the developerWorks archives, the video is no longer accessible.

The first contributing factor is that trade booking, back office, and other downstream systems often need drastic improvement and sometimes even a complete technology overhaul. These systems are slow, clunky, and, in some cases, not as secure as they should be, especially at larger banks. Developing new replacement software is a huge undertaking for any bank or investment fund, as...
their systems need to be robust enough to handle millions of data points a day and secure enough to comply with industry regulations.

The second factor is that even though the number of FinTech startups continues to increase and provide banks and investment firms with brand new, sophisticated technology, these startups typically tackle the "trendier" side of finance, like market data or predictive trading algos, leaving many areas in the finance industry underserved.

How distributed ledger technology can help

Distributed ledger technology, known as blockchain, enables all parties of a trade transaction to access the same source of information. This, in turn, helps streamline the order-to-settlement process and reduces the amount of extra work required to complete transactions. Hyperledger Fabric Version 1.0 is an open source blockchain framework and one of several Hyperledger projects hosted by The Linux Foundation. Because the latest release includes new open source developer tools, you can quickly and easily set up your own highly secure and private blockchain network in a matter of days.

Once a Hyperledger Fabric blockchain network is set up, it's governed by network policies that you create and manage, which allows for private communication and transactions between certain members of the group. This technology is extremely useful for almost any task or process involving sensitive data that a corporation or large business collects. It creates a solution for the three biggest problem areas for legacy financial network systems today: efficiency, cost, and security.

Use case: The trade lifecycle

Take the example of a bank taking orders from clients for trades. At each step of the trade lifecycle — from the client's trade request, all the way through to the cash actually changing hands — data management is essential. And because client and financial information is involved, this data must be secure and private.

Steps in the trade lifecycle:

1. Trade order placement
2. Trade capture and risk management
3. Trade confirmation and allocation
4. Exchange
5. Clearing and settlements

1. Trade order placement

Due to regulations resulting from the 2008 financial crisis, banks are not allowed to engage in proprietary trading, which essentially means that they are not allowed to make trades using their own money. All of their trades must be orders placed from or on behalf of their clients. This means that a bank needs to have a way to manage their client relationships and interactions.

A response to the 2008 financial crisis, the Dodd-Frank Wall Street Reform and Consumer Protection Act of 2010 mandated that banks are not to engage in proprietary ("prop") trading,
essentially meaning that they are not allowed to make trades using their own money. However, this Act has recently been repealed by the House, but it will take many months or even years for this change to be enacted. Additionally, banks still have the option to uphold their commitment to not engage in prop trading. Those banks that choose to not prop trade or choose to still have large deal flow from clients will need to have a way to manage their client relationships and interactions.

To do this, banks often use Client Relationship Management (CRM) systems, which are large, typically subscription-based applications. Banks rely on CRMs to keep their client's personal details private, not only to protect the clients, but also the reputation of the bank. Banks commonly choose to purchase a subscription to a CRM application like Salesforce or NexJ, but these can be expensive for software that just provides one function, and might not be as customizable and easily integrated with other bank technologies as a bank would like them to be.

Building a proprietary CRM system using Hyperledger Fabric would provide a network that is secure enough that banks can be assured their client information won't be exposed. Plus, it's easy to set up such a network using JavaScript, the most popular programming language used today. A bank could easily customize their proprietary Hyperledger Fabric-based CRM to fit the needs of their sales traders and clients in order to keep their old clients happy as well as to attract new clients.

**Example**

This sample shows how you could use a Hyperledger Fabric blockchain to maintain a database of clients, employees, and their messages and orders.

**Database type:** Identity management

**Some attributes of a user:**

- ID (unique string, to be used as a key)
- Name (string)
- Firm (string)
- Email Address (string)
- Office Location (string)
- Primary Relationship (string)
- User Type (bool)
- Primary Contact (string)

**Actions:**

- Send Message
- Place Order

**Action restrictions:**

- Employees can send a message to their client
- Clients can send messages to and place orders with their primary contact
In chaincode and using JSON, a simple example of a client struct, a data structure, would look like this:

**Listing 1. Sample chaincode snippet for trade order placement**

```go
type client struct {
    Id           string `json:"id`.
    Name            string `json:"name`.
    Firm            string `json:"firm`.
    Email           string `json:"email`.
    OfficeLocation  string `json:"officeLocation`.
    OrderHistory    string `json:"orderHistory`.
    PrimaryContact  string `json:"primContact`.
    PrimaryContactId string `json:"primContactID`.
}
```

The next step would be to write a function, such as `sendMessage`, to send a message between a client and an employee. To do this, take the attributes of `struct Client` as an array or as a parameter and set them equal to variables, along with the intended recipient ID. After checking if the intended recipient ID is equal to the primary contact ID, you can go ahead and authorize the sending of the message. A similar function can be written for the placement of an order.

### 2. Trade capture and risk management

When the order is placed by a client, the sales traders will send the trade details in a term sheet to their traders, who will enter the trade into the front office systems and perform risk management. This risk management includes a number of checks and calculations to determine the level of risk involved with the client’s order and whether it’s safe to be placed or if it needs to be hedged. Some of these checks involve determining whether the client has enough money to afford the security.

The systems involved in capturing these trade details and client account information could easily be enhanced using blockchain technology. Building a front-office trade capture system using Hyperledger Fabric is easy to do and would modernize most of the legacy systems that currently exist in most banks. It would make them more secure and able to more quickly and efficiently send data to other members of the team. For example, one member of the team could enter the trade details while another designated member could view those trade details and calculate risk and PV impact of those new orders. This information could then flow further downstream, into the trade confirmation and allocation systems.

There is often a distinction between front office and back office systems. Once the trade is captured in the front office systems by the traders or trading assistants, the risk and PV are calculated and approved of. Then the trade will need to enter the back office systems, which are connected to even more downstream systems. These back office systems work better if they are connected to the front office trade capture systems that send their information downstream immediately upon receiving approval from the traders.

**Example**

To model this, you could create a database to manage trades and trade details, which could be shared by the traders and the trade capture team and can also feed to any other system downstream.
Database type: Asset management

Some attributes of a trade:

- tradeID (unique string, to be used as a key)
- IsinCusip (string)
- Counterparty (string)
- Notional (int)
- Currency (string)
- DaysToSettle (int)
- ValuationModel (string)
- User (string)

Actions:

- Enter trade
- Update trade details

Action restrictions:

- Only trade capture team can enter trade details; all others have only read access

In chaincode, a simple example of a trade struct would look like this:

Listing 2. Sample chaincode snippet for trade capture and risk management

```go
type trade struct {
    tradeID     string `json:"tradeId"
    IsinCusip   string `json:"isinCusip"
    Counterparty string `json:"cpty"
    Notional    string `json:"notional"
    Currency    string `json:"ccy"
    DaysToSettle string `json:"daysToSettle"
    ValuationModel string `json:"ValuationModel"
    User        string `json:"user"
}
```

Here, you might want to check that the user is an authorized user before allowing the trade details to be entered or changed. Then, allow each of the param variables you set to be updated by the user.

3. Trade confirmation and allocation

Once the trade details are captured, the back office system feeds down into other systems that generate a trade confirmation, a report of the trade details that is sent to regulatory authorities, and settlement instructions, which detail how and when the cash will actually be exchanged. Setting up a Hyperledger Fabric network and giving permissions to the back office trade confirm teams as well as compliance teams and regulatory authority allows the back office teams to quickly and securely disseminate required information. The faster the information is sent and viewed, the more quickly breaks and discrepancies can be addressed by the regulatory authorities and compliance
teams, and rectified by the back office teams. The settlement instructions could be set up the same way, by giving permissions to all parties involved with the settlement to a database of each trade and its relevant settlement instructions.

**Example**

This example is similar to the previous one, so I won't create a struct here, because trade details will be the same or similar. However, you might also include an attribute `UserType` to easily check what type of user someone viewing the system is and whether or not they can respond to upstream systems or change trade data. It might be useful to allow others in the back office to change specific data but not all of it.

4. **Exchange**

Let's continue with our example to show how useful Hyperledger Fabric can be. The front office systems can send out the trade order details into the exchange so that a counterparty match can be made and reported to all parties and then tracked. Hyperledger Fabric has already been *used by the London Stock Exchange*, but I believe all exchanges could greatly benefit by adopting Blockchain-based technology. Hyperledger Fabric, in particular, would be a great fit because it's made for large businesses and it's easy to get set up. On a Blockchain-based exchange, trades would flow through the system and be matched more quickly, which is especially advantageous for firms engaging in high frequency trading. Market data would also be reported faster, and consensus would make the entire system more efficient and fair.

**Example**

This will be modeled similarly to the previous steps in the trade lifecycle. However, here, you will need to add an attribute for buying or selling. These details need to match a Financial Information eXchange (FIX) protocol, which is the standard industry protocol for sending trade data.

**Database type**: Asset management

**Some attributes of a trade order**:

- `tradeID` (unique string, to be used as a key)
- `IsinCusip` (string)
- `BuySell` (char)
- `Notional` (int)
- `Currency` (string)
- `DaysToSettle` (int)

**Actions**:

- Matching of trade details for opposite sides, buy and sell
- Allow transfer of ownership of stock or other assets

**Action restrictions**: 
• Only the placers of the trade and the exchange members should be viewing the trade details of a single trade

5. Clearing and settlements

The last step in the trade lifecycle is the clearing and settlements of the physical security. Clearing is an important step, as it is an intermediary that facilitates the actual transactions between counterparties. A clearing house takes the opposite position for each side of a trade. For the counterparty buying the security, it acts as the seller, and for the counterparty selling the security, it acts as the buyer.

Clearing houses improve the efficiency of the markets and add stability to the financial industry by ensuring that the counterparties have enough money to afford the transaction and ensure the success of the settlement. Consensus from a system built on Hyperledger Fabric would allow clearinghouses like the Depository Trust & Clearing Corporation (DTCC) to view trade terms and settlement instructions from each counterparty, and reconcile these to ensure that trades are settled correctly in a way everyone agrees upon.

Example

Once there is a match, that match will need to be fed into the clearing blockchain where the two counterparties will need to be modeled, along with a few of the trade details. You also need a smart contract stating that both parties have enough cash to be able to afford to pay and that they legally agree to pay what they owe. Then, once there is a legal smart contract in place, these details can send data downstream to financial systems that will actually physically exchange the cash.

Database type: Smart contract

Some attributes of the trade:

• CusipIsin (string)
• CounterpartyA (string)
• CounterpartyB (string)
• A_AccountNumber (string)
• B_AccountNumber (string)
• Notional (int)
• Currency (string)
• DaysToSettle (int)

Actions:

• Determine who owes what by looking at the "Buy" or "Sell" attributes of the trade

Action restrictions:

• Neither of the counterparties should have any view into what happens here
You need to follow the Hyperledger Fabric's model for creating smart contracts. This instructional guide shows you how to set up a smart contract as well as best practices that are important to implement.

#### Try Bluemix free for 30 days

Take advantage of powerful Bluemix services and infrastructure offerings to build, deploy, and run your apps in the cloud. Give it a try!

### Summary

You've now seen how Hyperledger Fabric could be used in a trading institution in a secure and efficient manner. Hyperledger Fabric blockchain networks provide security and, at the same time, allow you to govern a process that works best with network policies that you manage. Give Hyperledger Fabric v1.0 a try.
Related topics

- IBM Blockchain is based on Hyperledger Fabric
- Blockchain for financial services
- Blockchain, the next disruptor for finance (white paper)
- Building a blockchain PoC in ten minutes using Hyperledger Composer
- IBM Blockchain Developer Center
- IBM Blockchain service on Bluemix
- Blockchain essentials (free course for developers)

© Copyright IBM Corporation 2017
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)