**Set-up**

1. Can do it on the hyperledger composer or hyperledger fabric
2. If the client already has an BNA file go to

* <https://ibm-blockchain.github.io/platform-deployment/>
* Eventually they will need to create a business network on the hyperledger composer environment

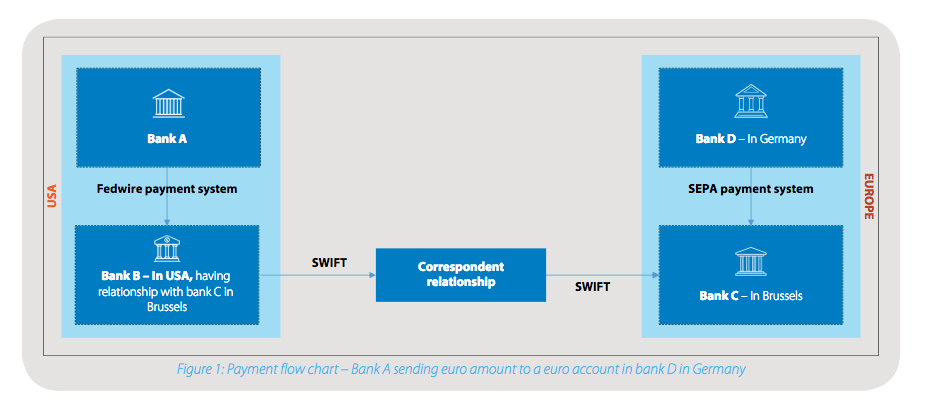
1. **If it is for demo or practice purposes use https://ibm-blockchain.github.io/setup/**

**Problem Statement**

The current payment infrastructure is mostly message based and relies on file exchange between multiple parties. A typical cross-border non-euro payment lifecycle involves several steps and parties to process the payments, which takes days to complete. Due to this delay, for example, a company initiating a payment is required to monitor its bank account or bank account statement to know exactly when the payment has cleared.

The problem occurs between banks that do not have an established financial relationship. Usually how it works is that money is executed through the Society for Worldwide Interbank Financial Telecommunication (SWIFT) network.

<https://www.infosys.com/industries/cards-and-payments/resources/Documents/cross-border-money-transfer.pdf>



**Objectives**

1. **Traceability (company)**
2. **Security (bank, company)**
3. **Processing speed (bank, company)**

**End-goal**

Blockchain’s features match the requirements for a payment infrastructure:

* Security
* Processing speed
* Traceability
* Global registry (ledger)

The distributed ledger makes it possible to connect all the parties in a financial trade in **real time for faster processing** of a payment – all while maintaining an **audit trail**.

If blockchain technology can be deployed on a large scale for payments, it would provide **significant benefits** for both banks and companies. Banks will no longer require heavy clearing and settlement systems, while companies can have a real-time accounting ledger in their enterprise resource planning (ERP) or treasury management systems. This would also improve payment security significantly.

**Hyperledger**

1. **[http:// 173.193.122.123:31080](http://184.172.229.53:31080) playground**
2. [**http://173.193.122.123:31090/explorer/**](http://173.193.122.123:31090/explorer/) **rest api server**
3. **When deploying a demo business network make sure to deploy it under hlfv1 NOT Web Browser.**
4. **When creating an REST server** 
   1. **“./create/create\_composer-rest-server.sh --paid --business-network-card INSERT\_BIZCARD\_NAME” when running this command replace “INSERT\_BIZCARD\_NAME” with the “admin@YOUR-EXAMPLE-NETWORK-NAME”**
   2. **Find external – IP using “kubectl get svc” under the “EXTERNAL-IP” category**
   3. **Access the REST server at “http://EXTERNAL-IP:31090/explorer/”**

**Participants**

Bank

* Have an original balance and its own database that’s not shared with anyone else.
* Can CREATE and UPDATE a transaction.
* Can view its own history transactions BOTH being the sender or the receiver .
* CANNOT send money to itself
* CANNOT access another banks database.
* CANNOT view other banks transactions.

Auditor

* Auditor is involved when a transaction is over 500,000.
  + The auditor will be randomly assigned from four auditing firms
* Can access its own history audits.
* Can only UPDATE the “auditor” and the “status” fields in trans.
* CANNOT read any transactions unless it is assigned to it.

Record Keeper

* Can READ all transactions

Regulator

* Can access when the status is set to be “FLAGGED”.
* Can READ when the price is over 1,000,000.
* Can only UPDATE the “regulator” and the “status” fields in trans.

**Asset**

* from (BankID)
* to(BankID)
* RK(RecordKeeperID)
* auditor(auditorID)
* regulator(regulatorID)
* transTime(transaction time)
* amount(the total amount of money for the transaction)
* status(the status of the transaction)

**Log**

May 17th

1. Is able to transfer money between banks
2. Banks cannot send money to themselves
3. STILL need to implement the check where a bank can only see its own transaction history.
4. The next step is to deploy it on Watson

May 18th

1. Framework is done
2. Need to come up with an example that is more relatable to the end consumers.

**Potential Problems and Tips**

1. When creating the access control logic its recommended to have the following three rules implemented. REMEMBER DO NOT DELETE the “NetworkAdminUser” and the “NetworkAdminSystem” rules as they the admin account to setup the whole blockchain.“AllAccess” rule would be useful during the testing stage, may need to modify the participant and the resource fields or add individual access rules for further testing.

rule NetworkAdminUser {

description: "Grant business network administrators full access to user resources"

participant: "org.hyperledger.composer.system.NetworkAdmin"

operation: ALL

resource: "\*\*"

action: ALLOW

}

rule NetworkAdminSystem {

description: "Grant business network administrators full access to system resources"

participant: "org.hyperledger.composer.system.NetworkAdmin"

operation: ALL

resource: "org.hyperledger.composer.system.\*\*"

action: ALLOW

}

rule AllAccess {

description: "AllAccess - grant everything to everybody."

participant: "org.hyperledger.composer.system.Participant"

operation: ALL

resource: "org.hyperledger.composer.system.\*\*"

action: ALLOW

}

1. If a rule is not implemented it is assumed to be false, therefore DENY.