

TP objectives:

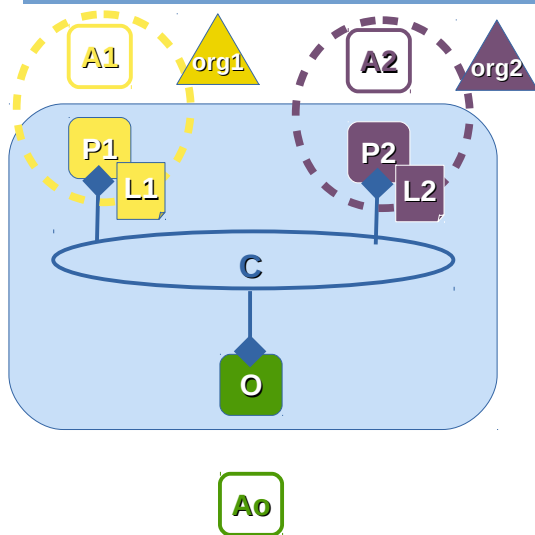
1/ Setup environment

2/ Configure Domain, and Security :

- Generate cryptographic material
- Generate genesis block
- Generate a channel

3/ Run network :

Network overview:



A1 & A2 : applications outside network

P1 : shop.sfeir.lu

P2 : warehouse.sfeir.lu

L1 & L2 : copy of ledger installed on peer P1 & P2

Org1 & Org2 : organization n*1, organization n* 2

O : oSfeir.sfeir.lu

C : channel : « sfeirchannel »

TP:

1. Setup environment

- Run command [1] ;

```
[1]  
./init.sh binaries
```

- Script should download **hyperledger fabric tools** and create **following directories**:

/bin	[hyperledger fabric tools]
-configtxgen	[generate channel configuration]
-configtxlator	[decrypt/encrypt messages]
-cryptogen	[generate crypto material]
-discover	[--]
-fabric-ca-client	[client to interact with fabric CA]
-idemixgen	[tool to build MSP directories]
-orderer	[execute cmd on orderer]
-peer	[execute cmd on peer]
/config	
-configtx.yaml	[domains configuration]
-core.yaml	[global peer configuration]
-orderer.yaml	[global network configuration]

2. Configure Domain, and Security

- Define and generate MSP configuration

Objectives :

- Define and generate cryptographic material for all network components of Blockchain.
(orderer, peers ...)

- Execute this command [1] :

```
[1]  
./init.sh setup
```

- Edit « crypto-config » file and check the configuration for expecting organisations, see section as below :

```
OrdererOrgs:
  - Name: Orderer
    Domain: sfeir.lu
  Specs:
    - Hostname: orderer
PeerOrgs:
  - Name: Shop
    Domain: shop.sfeir.lu
  Template:
    Count: 1
  Users:
    Count: 1
  - Name: Warehouse
    Domain: warehouse.sfeir.lu
  Template:
    Count: 1
  Users:
    Count: 1
```

- Execute this command [1] to generate the crypto material :

```
[1]
cryptogen generate --config=./crypto-config.yaml
```

- Check the result of command [1] :

```
/crypto-config
  (1) /ordererOrganizations
      /sfeir.lu
          /ca (1.1)
          |- ca.sfeir.lu-cert.pem
          |- <private key>
          /msp (1.2)
          /admincerts
          |- Admin@sfeir.lu-cert.pem
          /cacerts
          |- ca.sfeir.lu-cert.pem
          /tlscerts
          |- tlsca.sfeir.lu-cert.pem
      /orderers (1.3)
          /oSfeir.sfeir.lu
              /msp
                  /admincerts
                  |- Admin@sfeir.lu-cert.pem
                  /cacerts
                  |- ca.sfeir.lu-cert.pem
                  /keystore
                  |- <private key>
                  /signcerts
                  |- oSfeir.sfeir.lu-cert.pem
                  /tlscacerts
```

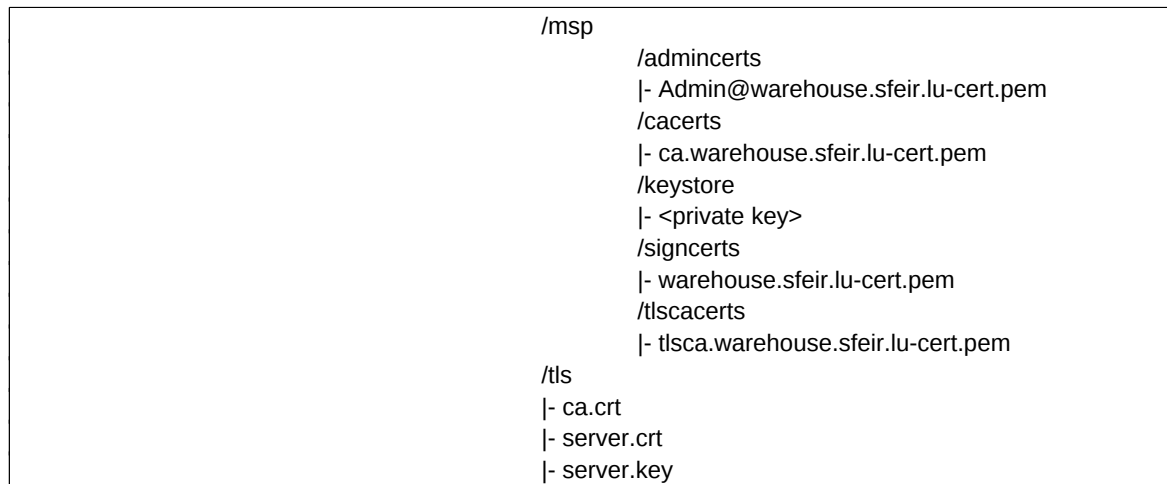
```

| - tlsca.sfeir.lu-cert.pem
/tls
| - ca.crt
| - server.crt
| - server.key
/tlsca
| - tlsca.sfeir.lu-cert.pem
| - <private key>
/users (1.5)
  /Admin@sfeir.lu
    /admincerts
    | - Admin@sfeir.lu-cert.pem
    /cacerts
    | - ca.sfeir.lu-cert.pem
    /keystore
    | - <private key>
    /signcerts
    | - oSfeir.sfeir.lu-cert.pem
    /tlscacerts
    | - tlsca.sfeir.lu-cert.pem
/tls
| - ca.crt
| - server.crt
| - server.key
(2) /peerOrganizations
  /peer0.shop.sfeir.lu
  /peer0.warehouse.sfeir.lu
    /ca (2.1)
    | - ca.warehouse.sfeir.lu-cert.pem
    | - <private key>
    /msp (2.2)
    | /admincerts
    | | - Admin@warehouse .sfeir.lu-cert.pem
    | /cacerts
    | | - ca.warehouse.sfeir.lu-cert.pem
    | /tlscerts
    | | - tlsca.warehouse.sfeir.lu-cert.pem
  /peers (2.3)
  | /msp
  | | /admincerts
  | | | - Admin@warehouse .sfeir.lu-cert.pem
  | | /cacerts
  | | | - ca.warehouse.sfeir.lu-cert.pem
  | | /keystore
  | | | - <private key>
  | | /signcerts
  | | | - peer0.warehouse.sfeir.lu-cert.pem
  | | /tlscacerts
  | | | - tlsca.warehouse.sfeir.lu-cert.pem
  /tls
  | - ca.crt
  | - server.crt
  | - server.key
/tlsca (2.4)
| - tlsca.sfeir.lu-cert.pem
| - <private key>
/users (2.5)
  /Admin@Warehouse.sfeir.lu

```

SIGN PROCESS

VERIFY PROCESS



(1) and (2) – crypto material configuration for each **organization**.

(1.1) and (2.1) - crypto material for the **Root Certification Authority (CA) of organization** to deliver certificates for CA intermediaries which participate to BC process.

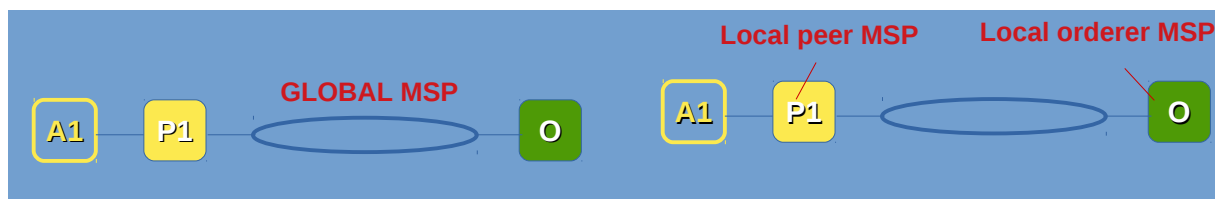
(1.2) and (2.2) - crypto material for the **Membership Service Provider of organization** in charge to deliver verifiable member identities.

(1.3) and (2.3) - specific crypto material for each membership of the organization issue of root CA, MSP, TLS ...

(1.4) and (2.4) - crypto material for the **Root Certification Authority of organization** to deliver certificates for **TLS transport**.

(1.5) and (2.5) – crypto material for each users of organization (administrator, specific users...)

MSP is installed at different level of the network :



notes : Global configuration will corresponds to 1.1 (2.1), 1.2 (2.2), 1.4 (2.4), 1.5 (2.5).

Each **Organization** requires a unique **root certificate** (ca), that binds peers and orderers.

Transactions and **communications** (tlscert) within Fabric are **signed** by an entity's **private key** (keystore), and then **verified** by means of a **public key** (signcerts used for authenticating).

Some complexe organization needs to certifiante peers on their organization, and need a CA (intermediate) for that.

3. Generate genesis block

a) Define ledger configuration

- Edit file «**configtx.yaml**», file and check the configuration for expecting organisations, see section as below as below :

```
Organizations:
- &OrdererOrg
  Name: OrdererOrg
  ID: OSfeirMSP
  MSPDir: crypto-config/ordererOrganizations/sfeir.lu/msp
- &Org1
  Name: ShopMSP
  ID: ShopMSP
  MSPDir: crypto-config/peerOrganizations/shop.sfeir.lu/msp
  AnchorPeers:
    - Host: peer0.shop.sfeir.lu
      Port: 7051
- &Org2
  Name: WarehouseMSP
  ID: WarehouseMSP
  MSPDir: crypto-config/peerOrganizations/warehouse.sfeir.lu/msp
  AnchorPeers:
    - Host: peer0.warehouse.sfeir.lu
      Port: 7051
#####
Application: &ApplicationDefaults
  Organizations:
#####
Orderer: &OrdererDefaults
  OrdererType: solo
  Addresses:
    - orderer.sfeir.lu:7050
  BatchTimeout: 2s
  BatchSize:
    MaxMessageCount: 10
    AbsoluteMaxBytes: 99 MB
    PreferredMaxBytes: 512 KB
  Kafka:
    Brokers:
      - 127.0.0.1:9092
  Organizations:
#####
Profiles:
  OrdererGenesis:
    Orderer:
      <<: *OrdererDefaults
    Organizations:
      - *OrdererOrg
    Consortiums:
      SampleConsortium:
        Organizations:
          - *Org1
          - *Org2
  OneOrgChannel:
    Consortium: SampleConsortium
    Application:
      <<: *ApplicationDefaults
    Organizations:
      - *Org1
```

```
- *Org2
```

b) Generate first block of ledger

- Execute this command [1] to generate first block :

```
[1] #  
configtxgen -profile OrdererGenesis -outputBlock ./config/genesis.block
```

- Check that a new file is generated issue of command [1]. (on directory ./channel-artifacts)

```
[common/tools/configtxgen] doOutputBlock -> INFO 007 Generating genesis block  
[common/tools/configtxgen] doOutputBlock -> INFO 008 Writing genesis block
```

4. Generate channel + configuration

- Execute this command [1] :

```
[1] #  
configtxgen -profile TwoOrgChannel -outputCreateChannelTx ./config/channel.tx -channelID sfeircn
```

- A new channel « sfeircn » is created.

```
[common/tools/configtxgen] doOutputChannelCreateTx -> INFO 006 Writing new channel tx
```

- Execute this command [2] and [3] :

```
[2] #  
configtxgen -profile TwoOrgChannel -outputAnchorPeersUpdate ./config/ShopMSPanchors.tx  
-channelID sfeircn -asOrg ShopMSP
```

```
[3] #  
configtxgen -profile TwoOrgChannel -outputAnchorPeersUpdate ./config/WarehouseMSPanchors.tx  
-channelID sfeircn -asOrg WarehouseMSP
```

- Shop and Warehouse org are added to the channel « sfeircn ».

Command [2] or [3] results :

```
[common/tools/configtxgen] doOutputAnchorPeersUpdate -> INFO 002 Generating anchor peer update  
[common/tools/configtxgen] doOutputAnchorPeersUpdate -> INFO 003 Writing anchor peer update
```

5. Run network

- Run command [1] ;

```
[1] #  
  
./init.sh docker-images
```

- Script will download docker hyperledger-fabric images inside your repositories. You can then check on your repositories: (it can get time to fetch images ...)

```
[1] #  
  
docker images  
  
==>  
REPOSITORY          TAG          IMAGE ID      CREATED      SIZE  
hyperledger/fabric-tools  1.3.0       7aea1cc899d1  9 days ago  1.51GB  
hyperledger/fabric-tools  latest      7aea1cc899d1  9 days ago  1.51GB  
hyperledger/fabric-ccenv  1.3.0       8651e7160d88  9 days ago  1.43GB  
hyperledger/fabric-ccenv  latest      8651e7160d88  9 days ago  1.43GB  
hyperledger/fabric-orderer 1.3.0       b1a1dd788841  9 days ago  152MB  
hyperledger/fabric-orderer latest      b1a1dd788841  9 days ago  152MB  
hyperledger/fabric-peer    1.3.0       ef0e7788ead0  9 days ago  159MB  
hyperledger/fabric-peer    latest      ef0e7788ead0  9 days ago  159MB  
hyperledger/fabric-ca      1.3.0       784b38dab5ba  11 days ago  244MB  
hyperledger/fabric-ca      latest      784b38dab5ba  11 days ago  244MB  
hyperledger/fabric-zookeeper 0.4.12     bca71b814159  2 weeks ago  1.39GB  
hyperledger/fabric-zookeeper latest      bca71b814159  2 weeks ago  1.39GB  
hyperledger/fabric-kafka   0.4.12     58b901c762ea  2 weeks ago  1.4GB  
hyperledger/fabric-kafka   latest      58b901c762ea  2 weeks ago  1.4GB  
hyperledger/fabric-couchdb 0.4.12     fe8d64d1233c  2 weeks ago  1.45GB  
hyperledger/fabric-couchdb latest      fe8d64d1233c  2 weeks ago  1.45GB
```

- Run command [2] ;

```
[2] #  
  
./init.sh replace-pki
```

Notes : edit « docker-sfeir.yaml » to consult docker compose configuration.

```
[3] #  
  
docker-compose -f docker-sfeir.yaml up --remove-orphans -d ca.shop.sfeir.lu  
ca.warehouse.sfeir.lu orderer.sfeir.lu peer0.shop.sfeir.lu peer0.warehouse.sfeir.lu
```

- Command [3] starts ca(s), orderer, couchdb and peer(s) containers.

Notes : couchdb is used to store ledger metadata data.


```
Starting ca.shop.sfeir.lu  
Starting orderer.sfeir.lu  
Starting ca.warehouse.sfeir.lu  
Starting couchdb  
Starting peer0.shop.sfeir.lu  
Starting peer0.warehouse.sfeir.lu
```

6. Install and instanciate channel

- Run command [1] ;

```
[1] #  
  
docker exec -e "CORE_PEER_LOCALMSPID=ShopMSP" -e  
"CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@shop.sfeir.lu/msp"  
peer0.shop.sfeir.lu peer channel create -o orderer.sfeir.lu:7050 -c sfeircn -f  
/etc/hyperledger/configtx/channel.tx
```

- check results on log :
« readBlock -> INFO 004 Received block: 0 »
- Run command [2] ;

```
[2] #  
  
docker exec -e "CORE_PEER_LOCALMSPID=ShopMSP" -e  
"CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@shop.sfeir.lu/msp"  
peer0.shop.sfeir.lu peer channel join -b sfeircn.block
```

- check results on log :
« [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel »
- Run command [3] ;

```
[3] #  
  
docker exec peer0.warehouse.sfeir.lu peer channel fetch 0 sfeircn.block --channelID sfeircn --orderer  
orderer.sfeir.lu:7050
```

- check results on log :
« readBlock -> INFO 002 Received block: 0 »
- Run command [4] ;

```
[4] #  
  
docker exec -e "CORE_PEER_LOCALMSPID=WarehouseMSP" -e  
"CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@warehouse.sfeir.lu  
/msp" peer0.warehouse.sfeir.lu peer channel join -b sfeircn.block
```

- check results on log :
« [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel »

7. Install Chaincode

- Run command [1] ;

```
[1] #  
  
docker-compose -f docker-sfeir.yaml up -d cli
```

- Run command [2] to install the chaincode in channel for peer 1 ;

```
[2] #  
  
docker exec cli peer chaincode install -n sfeircc -v 1.0 -l golang -p  
github.com/hyperledger/fabric/examples/chaincode/go
```

- check results on log :

```
2018-10-13 07:09:26.321 UTC [chaincodeCmd] install -> INFO 052 Installed remotely response:<status:200 payload:"OK" >
```

- Run command [3] to install the chaincode in channel for peer 2 ;

```
[3] #  
  
docker exec -e "CORE_PEER_ADDRESS=peer0.warehouse.sfeir.lu:7051" -e  
"CORE_PEER_LOCALMSPID=WarehouseMSP" -e  
"CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/warehouse.sfeir.lu/users/Admin@warehouse.sfeir.lu/msp" cli peer chaincode  
install -n sfeircc -v 1.0 -l golang -p github.com/hyperledger/fabric/examples/chaincode/go
```

- check results on log :

```
2018-10-13 07:09:26.321 UTC [chaincodeCmd] install -> INFO 052 Installed remotely response:<status:200 payload:"OK" >
```

8. Invoke Chaincode

- Run command [1] ;

[1] #

```
docker exec cli peer chaincode instantiate -o orderer.sfeir.lu:7050 -C sfeircn -n sfeircc -l golang -v 1.0 -c '{"Args":["init"]}'
```

```
docker exec cli peer chaincode instantiate -o orderer.sfeir.lu:7050 -C sfeircn -n sfeircc -l golang -v 1.0 -c '{"Args":["init", "0001", "SHOP_1","SKU002","10000", "1000"]}'
```

```
docker exec cli peer chaincode invoke -o orderer.sfeir.lu:7050 -C sfeircn -n sfeircc -c '{"Args":["read", "0001"]}'
```

```
docker exec cli peer chaincode query -C sfeircn -n sfeircc -c '{"Args":["read", "0001"]}'
```

Notes :

MSP : « Membership Service Provider » : Allows to identify actor inside network, and to provide CA to trust member identity.

Crypto material : « [crypto-config.yaml](#) » file ;

- "OrdererOrgs" - Definition of organizations managing orderer nodes.
 - Define a name, and the business domain for an ordererOrgs.
 - Use « Specs » for an explicit definition of peer : hostname (+ commonName)
 - (or) Use «Template » for quickly create peer sequentially.
- "PeerOrgs" - Definition of organizations managing peer nodes.