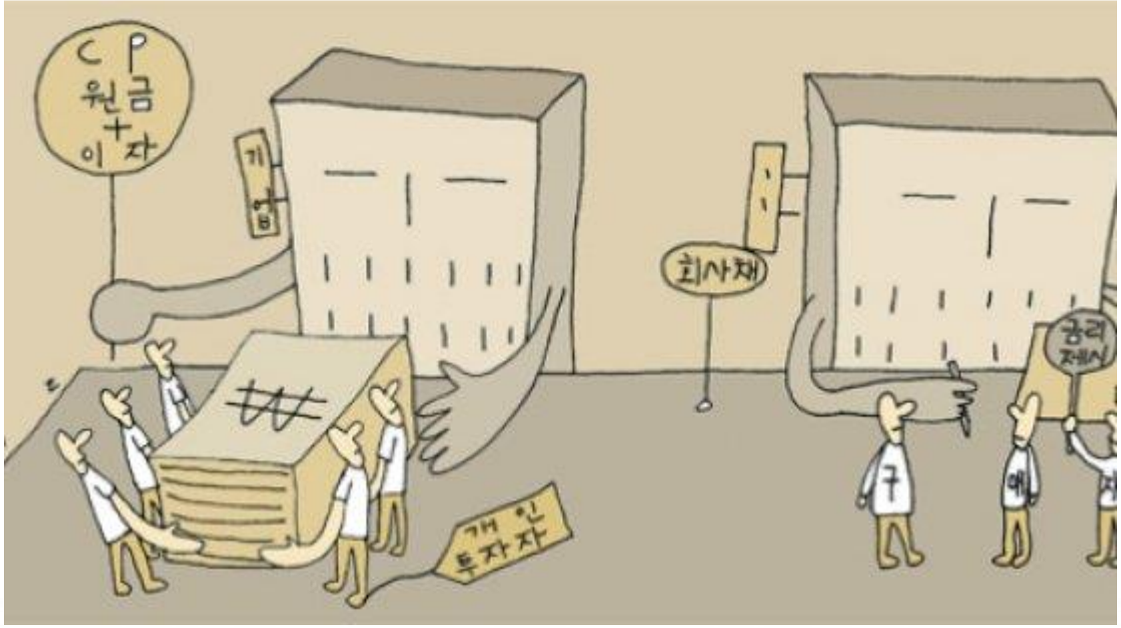
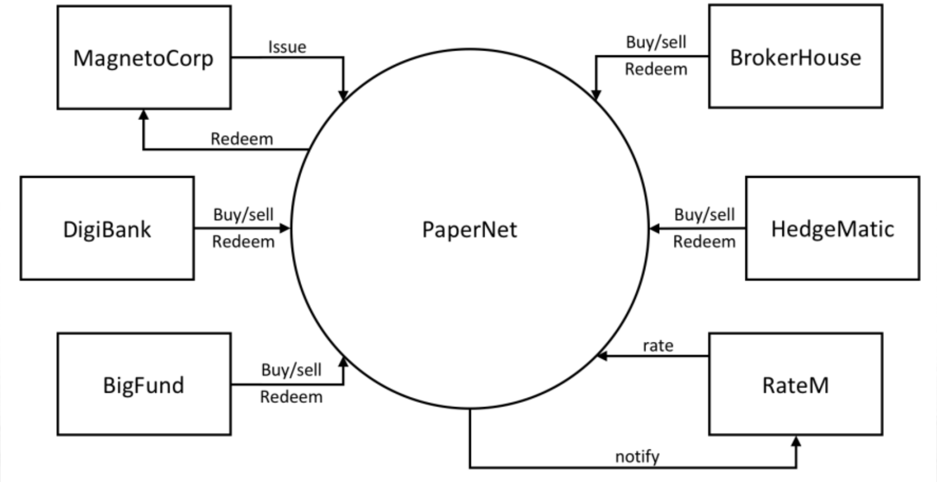
비지니스 네트워크 작성예제



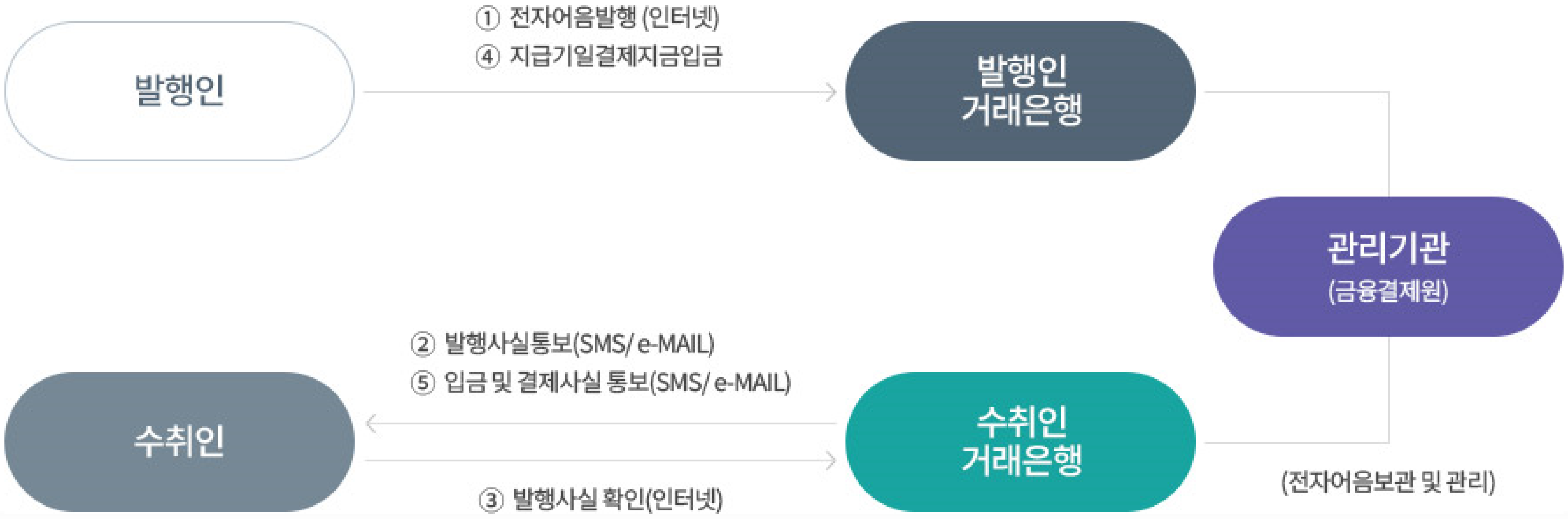
비즈니스 시나리오

금융상품(어음) 거래 네트워크 (PaperNet commercial paper network)

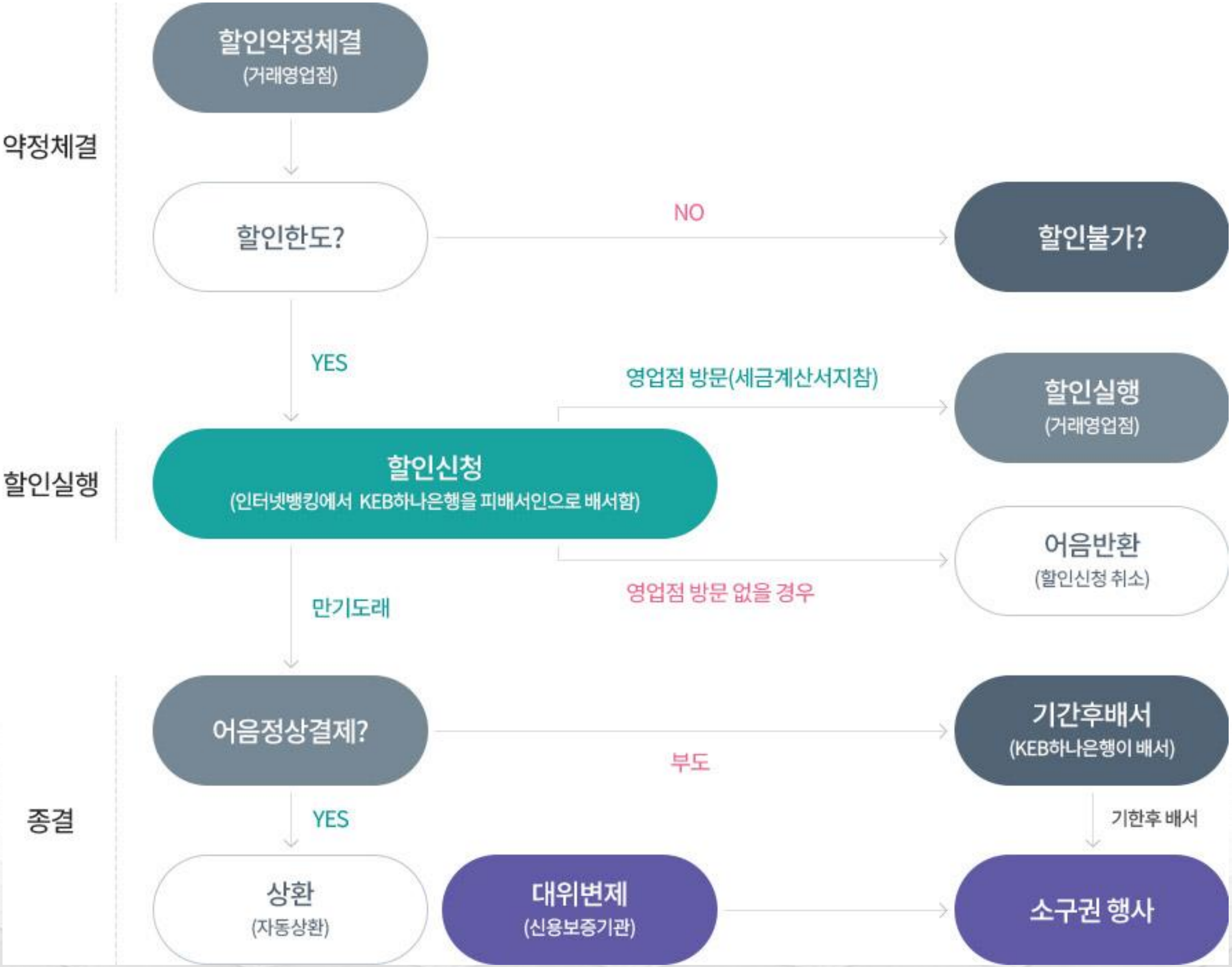
- 6개의 기관이 어음 발행, 구매, 판매, 상환(회수), 시세조정을 위하여 금융 네트워크(PaperNet)를 사용함
- MagentoCorp issues and redeems commercial paper.
- DigiBank, BigFund, BrokerHouse and HedgeMatic all trade commercial paper with each other.
- RateM provides various measures of risk for commercial paper.



전자어음-하나은행


① 전자어음발행 (인터넷)
④ 지급기일결제지급입금
발행인
발행인 거래은행
관리기관
(금융결제원)
② 발행사실통보(SMS/ e-MAIL)
⑤ 입금 및 결제사실 통보(SMS/ e-MAIL)
수취인
수취인 거래은행
③ 발행사실 확인(인터넷)
(전자어음보관 및 관리)

전자어음-하나은행



약정체결

할인약정체결
(거래영업점)

할인한도?

NO

할인불가?

YES

할인실행

할인신청
(인터넷뱅킹에서 KEB하나은행을 피배서인으로 배서함)

영업점 방문(세금계산서지참)

할인실행
(거래영업점)

영업점 방문 없을 경우

어음반환
(할인신청 취소)

만기도래

종결

어음정상결제?

부도

기간후배서
(KEB하나은행이 배서)

기한후 배서

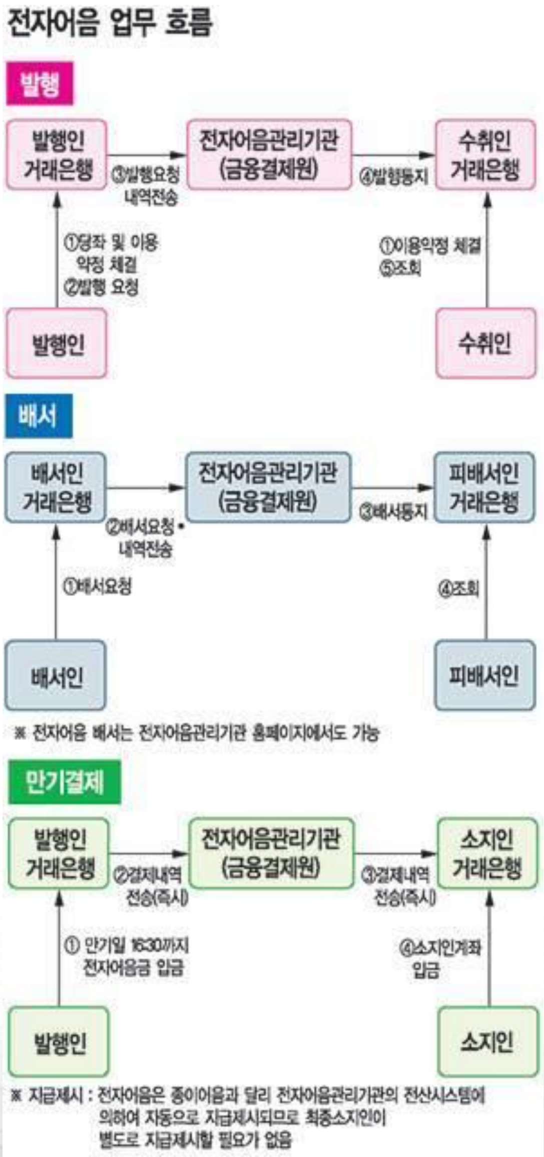
YES

상환
(자동상환)

대위변제
(신용보증기관)

소구권 행사

전자어음 업무흐름

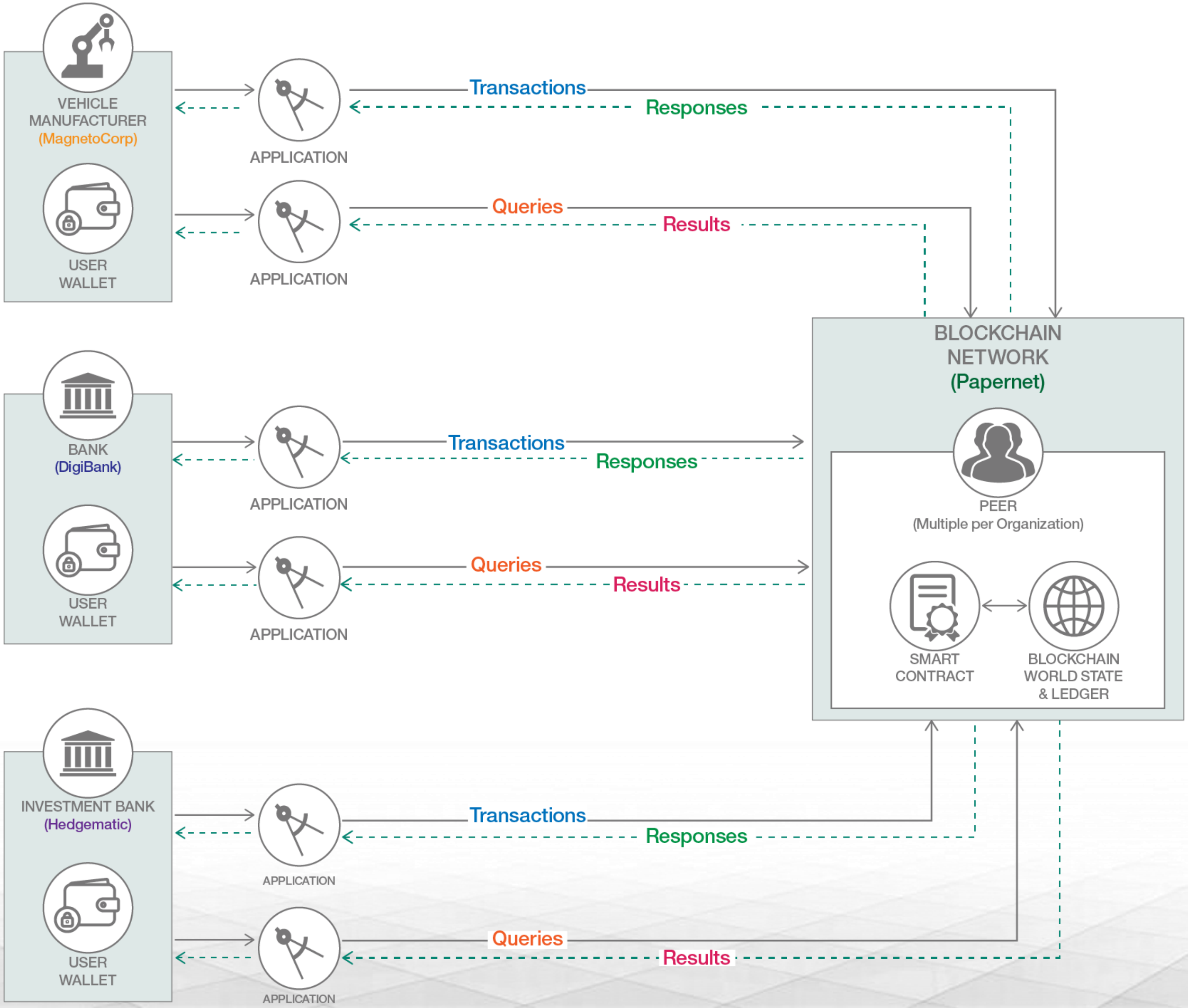


전자어음 업무 흐름

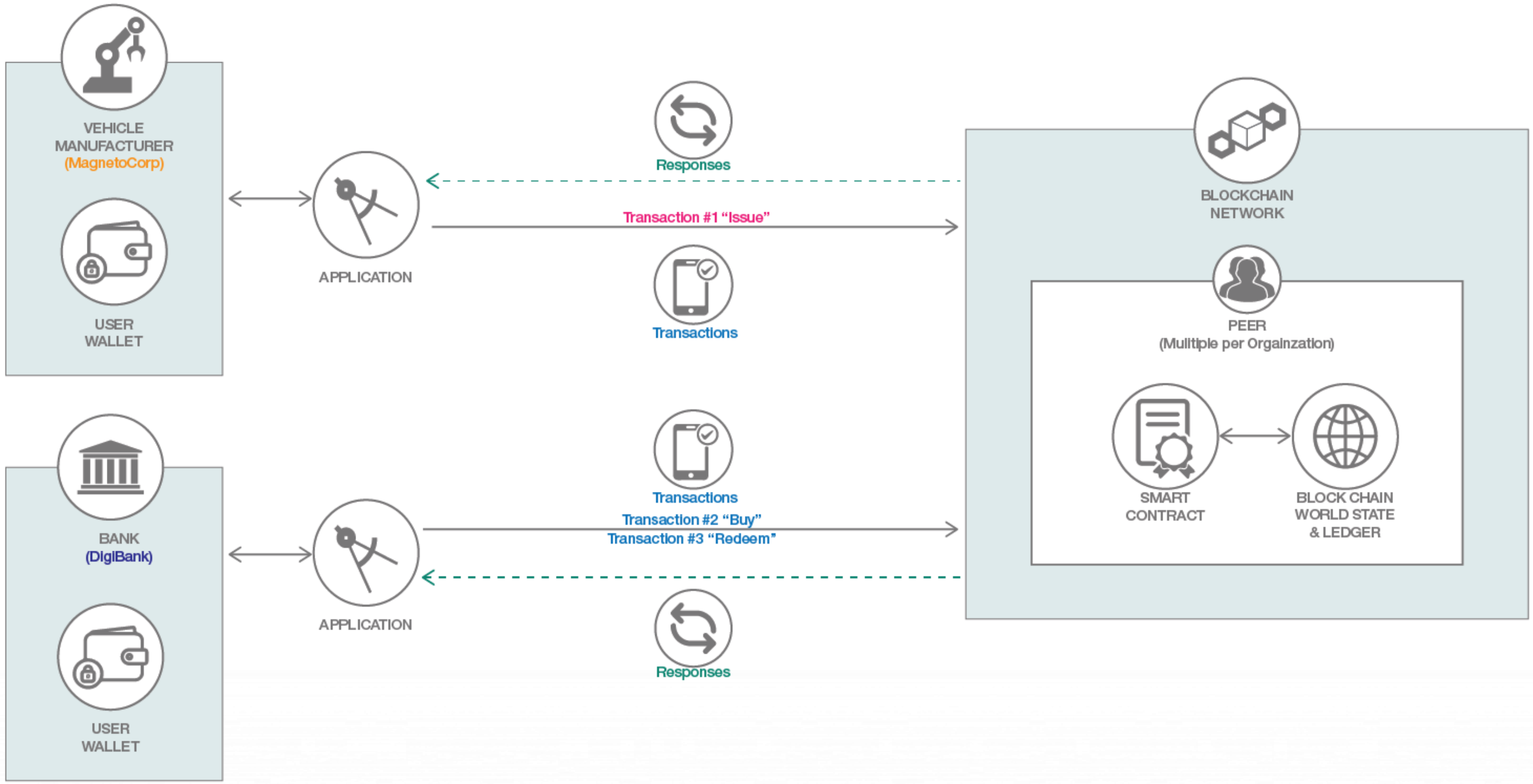
※ 전자어음 배서는 전자어음관리기관 홈페이지에서도 가능

※ 지급제시 : 전자어음은 종이어음과 달리 전자어음관리기관의 전산시스템에 의하여 자동으로 지급제시되므로 최종소지인이 별도로 지급제시할 필요가 없음

응용프로그램 연동구조


VEHICLE MANUFACTURER
(MagnetoCorp)
USER WALLET
APPLICATION
Transactions
Responses
APPLICATION
Queries
Results
BANK
(DigiBank)
USER WALLET
APPLICATION
Transactions
Responses
APPLICATION
Queries
Results
INVESTMENT BANK
(Hedgematic)
USER WALLET
APPLICATION
Transactions
Responses
APPLICATION
Queries
Results
BLOCKCHAIN NETWORK
(Papernet)
PEER
(Multiple per Organization)
SMART CONTRACT
BLOCKCHAIN WORLD STATE & LEDGER

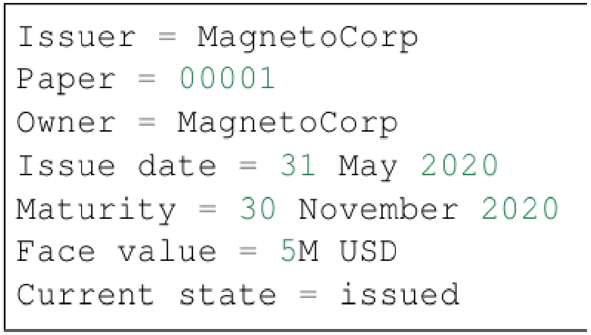
응용프로그램 연동구조


VEHICLE
MANUFACTURER
(MagnetoCorp)
USER
WALLET
APPLICATION
Responses
Transaction #1 "Issue"
Transactions
BLOCKCHAIN
NETWORK
PEER
(Mulitple per Orgainzation)
SMART
CONTRACT
BLOCK CHAIN
WORLD STATE
& LEDGER
BANK
(DigiBank)
USER
WALLET
APPLICATION
Transactions
Transaction #2 "Buy"
Transaction #3 "Redeem"
Responses

어음의 생명주기: 발행, 거래, 상환

비즈니스 분석

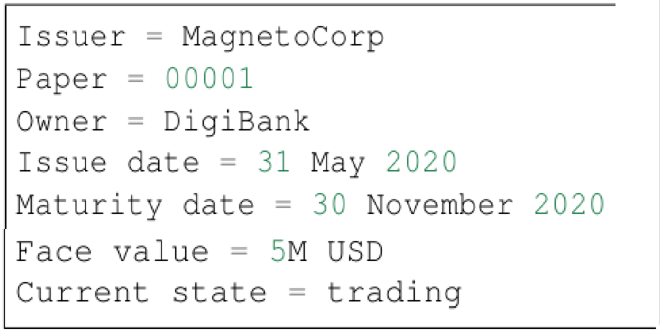
issue transaction
(first state of this paper)



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

MagnetoCorp이 어음을 발행

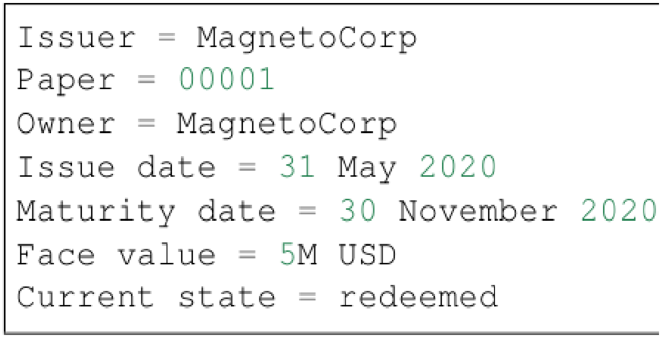
buy transaction



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

DigiBank가 어음을 구매

final redeem transaction



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

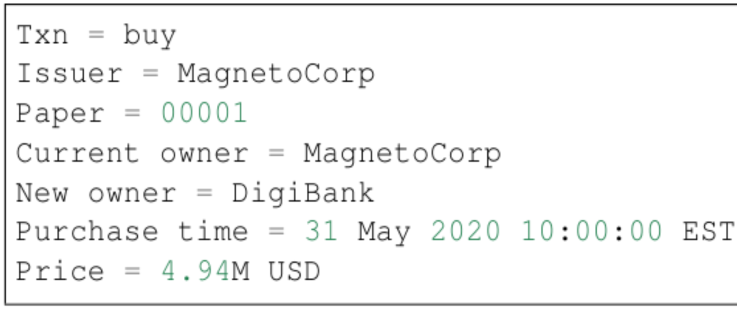
MagnetoCorp이 어음을 상환

트랜잭션

Transactions

1. Issue

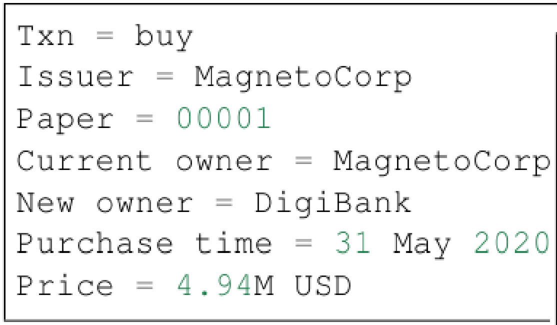
발행자 서명 필요



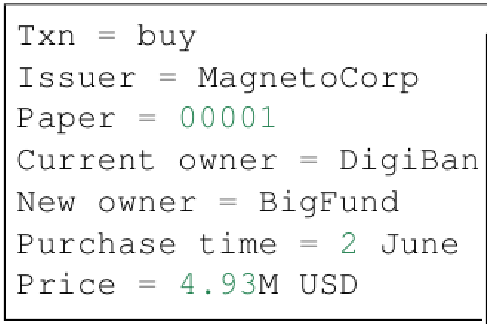
```
Txn = buy
Issuer = MagnetoCorp
Paper = 00001
Current owner = MagnetoCorp
New owner = DigiBank
Purchase time = 31 May 2020 10:00:00 EST
Price = 4.94M USD
```

2. buy

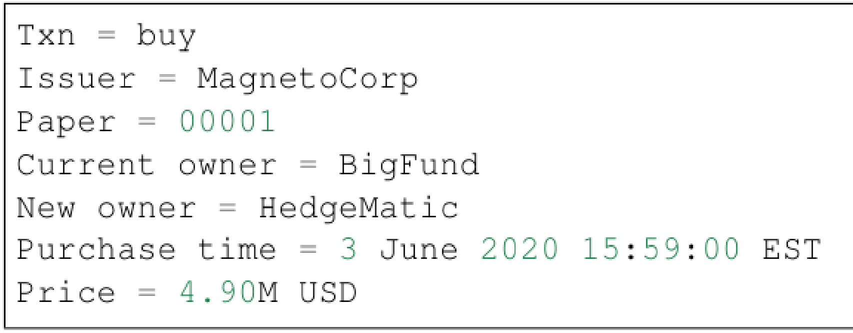
사고 파는 기관의 서명 필요



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

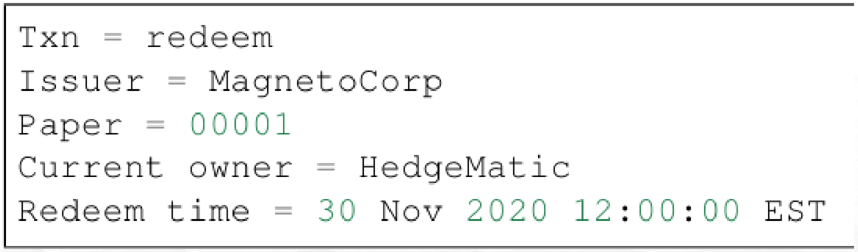


```
Txn = buy
Issuer = MagnetoCorp
Paper = 00001
Current owner = DigiBan
New owner = BigFund
Purchase time = 2 June
Price = 4.93M USD
```



```
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
```

3. Redeem 사고(상환) 파는 사람 서명 필요



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

장부

- Ledger
- Hyperledger Fabric distributed ledger
  - world state: 현재의 상태가 저장
  - Blockchain: 트랜잭션 히스토리 레코드 저장 (현재의 상태에 반영됨)
- 트랜잭션(거래)에서 서명은 지켜야 하는 룰이며, 서명이 되었나 검증됨
- 이러한 생각(비즈니스 로직)을 스마트컨트랙트(프로그래밍)로 변환해야함

138

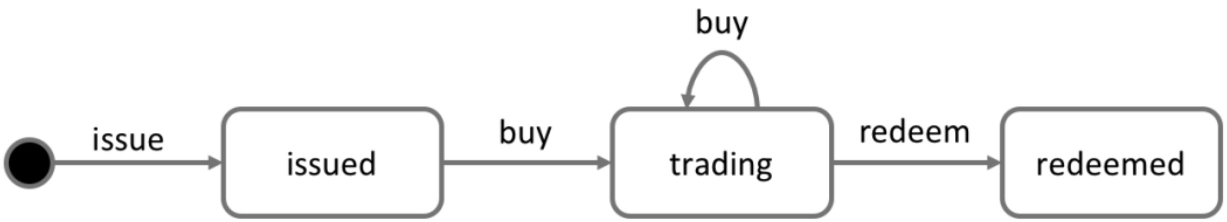
비즈니스 프로세스와 데이터 설계

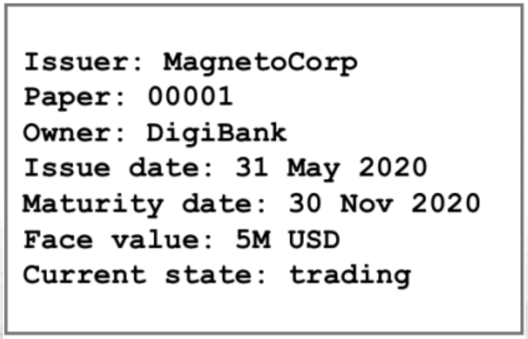
생명주기

- 상태전이도(state transition diagram): issued, trading, redeemed states

원장 상태(Ledger state)

- 어음은 일련의 속성(properties)과 값(value)으로 표현된다.
- 일반적으로 속성이 결합하여 어음마다 유일한 키(unique key)가 된다.





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

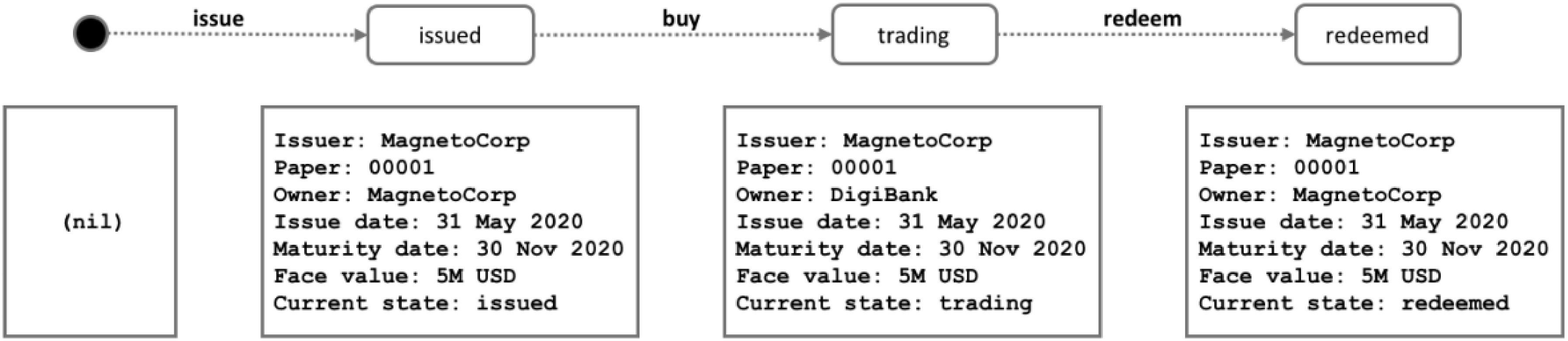
비즈니스 프로세스와 데이터 설계

원장 상태 (Ledger state)

- 트랜잭션에 의해 상태는 전이 됨 (벡터값)

상태키 (State keys)

- Key: 유일한 Id. (MagnetoCorp00001) - 발행자와 paper 속성값이 결합된다
.
- 원장에서 개별적인 상태 객체는 유일한 키값을 갖는다.



```
(nil)
```

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

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

```
Issuer: MagnetoCorp
Paper: 00001
Owner: MagnetoCorp
Issue date: 31 May 2020
Maturity date: 30 Nov 2020
Face value: 5M USD
Current state: redeemed
```

스마트컨트랙트 처리

Smart Contract Class

- Contract class and Context class were brought into cope:

```
const { Contract, Context } = require('fabric-contract-api');
```

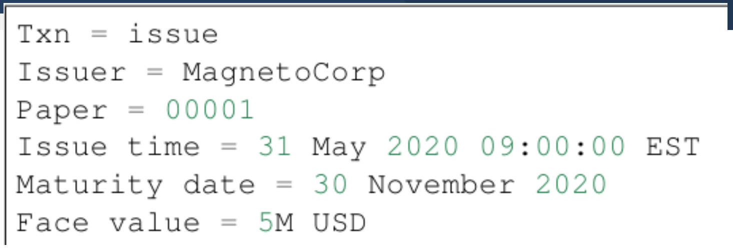
- CommercialPaperContract extends the Hyperledger Fabric Contract class
- (definition for the commercial paper smart contract):

```
class CommercialPaperContract extends Contract {...}
```

- Class constructor uses its superclass to initialize itself with an explicit contract name:

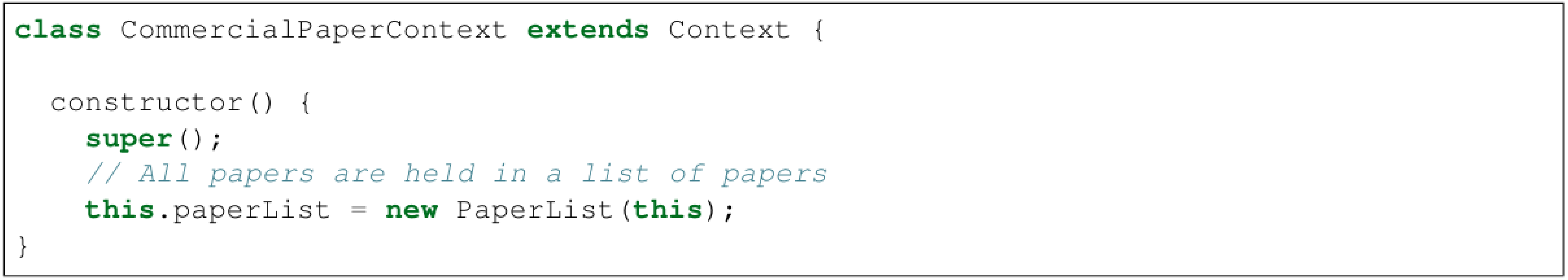
```
constructor() {
    super('org.papernet.commercialpaper');
}
```

트랜잭션 정의



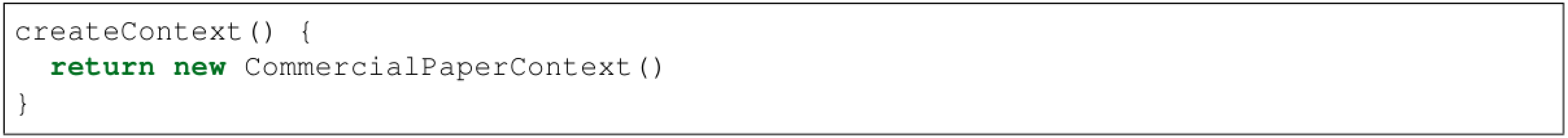
```
Txn = issue
Issuer = MagnetoCorp
Paper = 00001
Issue time = 31 May 2020 09:00:00 EST
Maturity date = 30 November 2020
Face value = 5M USD
```

- extended context adds a custom property paperList to the defaults:



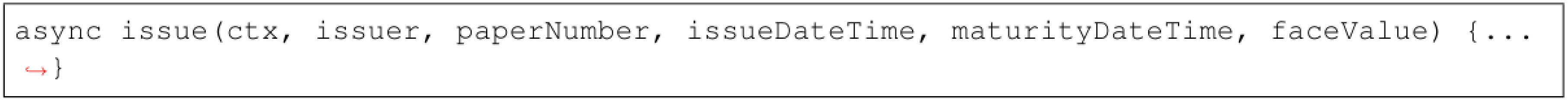
```
class CommercialPaperContext extends Context {

  constructor() {
    super();
    // All papers are held in a list of papers
    this.paperList = new PaperList(this);
}
```

- smart contract extends the default transaction context by implementing its own createContext()



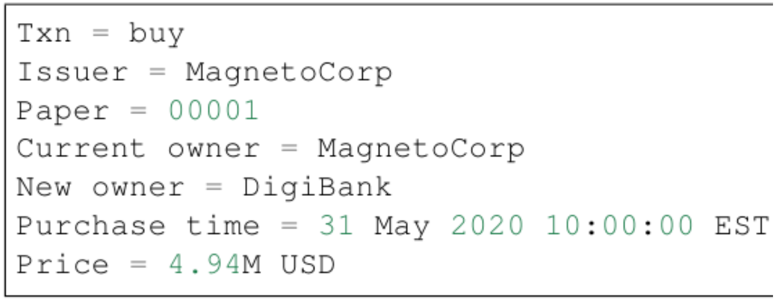
```
createContext() {
  return new CommercialPaperContext()
}
```

- Within the class, locate the issue method.



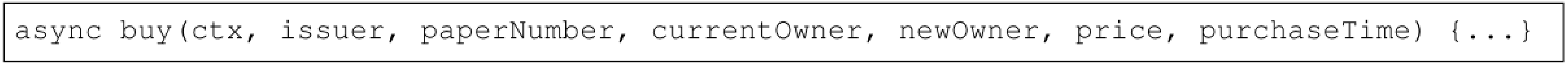
```
async issue(ctx, issuer, paperNumber, issueDateTime, maturityDateTime, faceValue) {...
 ↪}
```

트랜잭션 정의 2

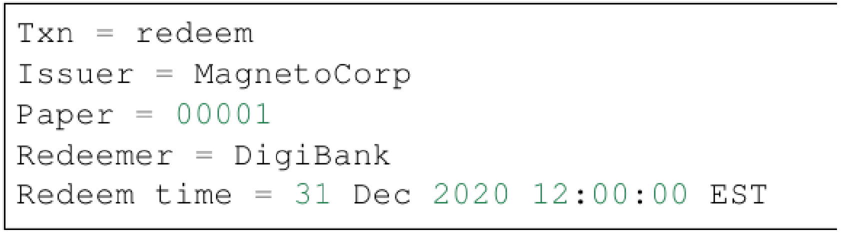


```
Txn = buy
Issuer = MagnetoCorp
Paper = 00001
Current owner = MagnetoCorp
New owner = DigiBank
Purchase time = 31 May 2020 10:00:00 EST
Price = 4.94M USD
```

buy transaction:

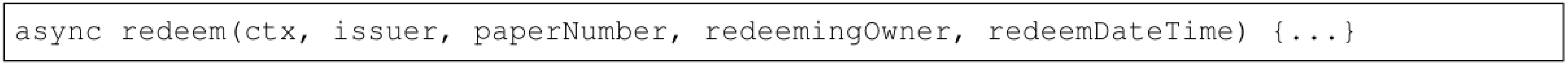


```
async buy(ctx, issuer, paperNumber, currentOwner, newOwner, price, purchaseTime) {...}
```



```
Txn = redeem
Issuer = MagnetoCorp
Paper = 00001
Redeemer = DigiBank
Redeem time = 31 Dec 2020 12:00:00 EST
```

redeem transaction:

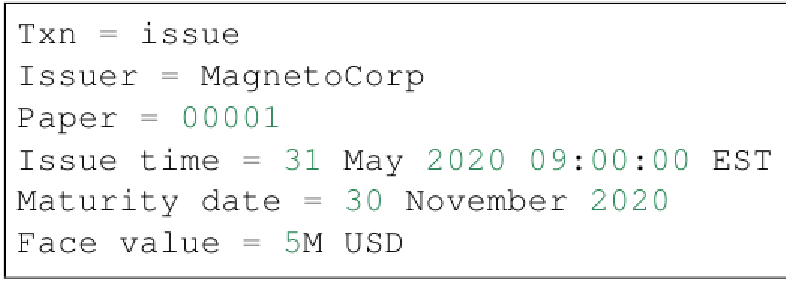


```
async redeem(ctx, issuer, paperNumber, redeemingOwner, redeemDateTime) {...}
```

143

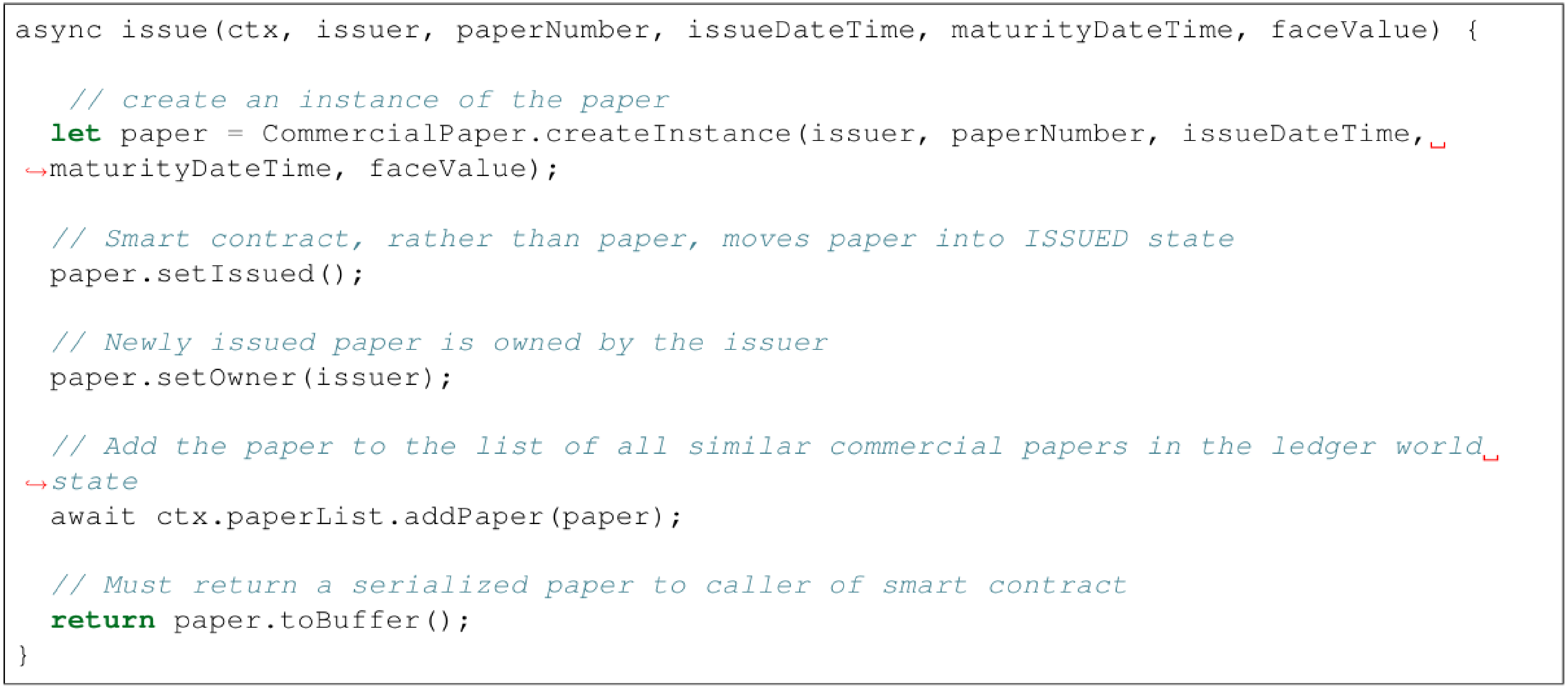
트랜잭션 로직

issue transaction



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

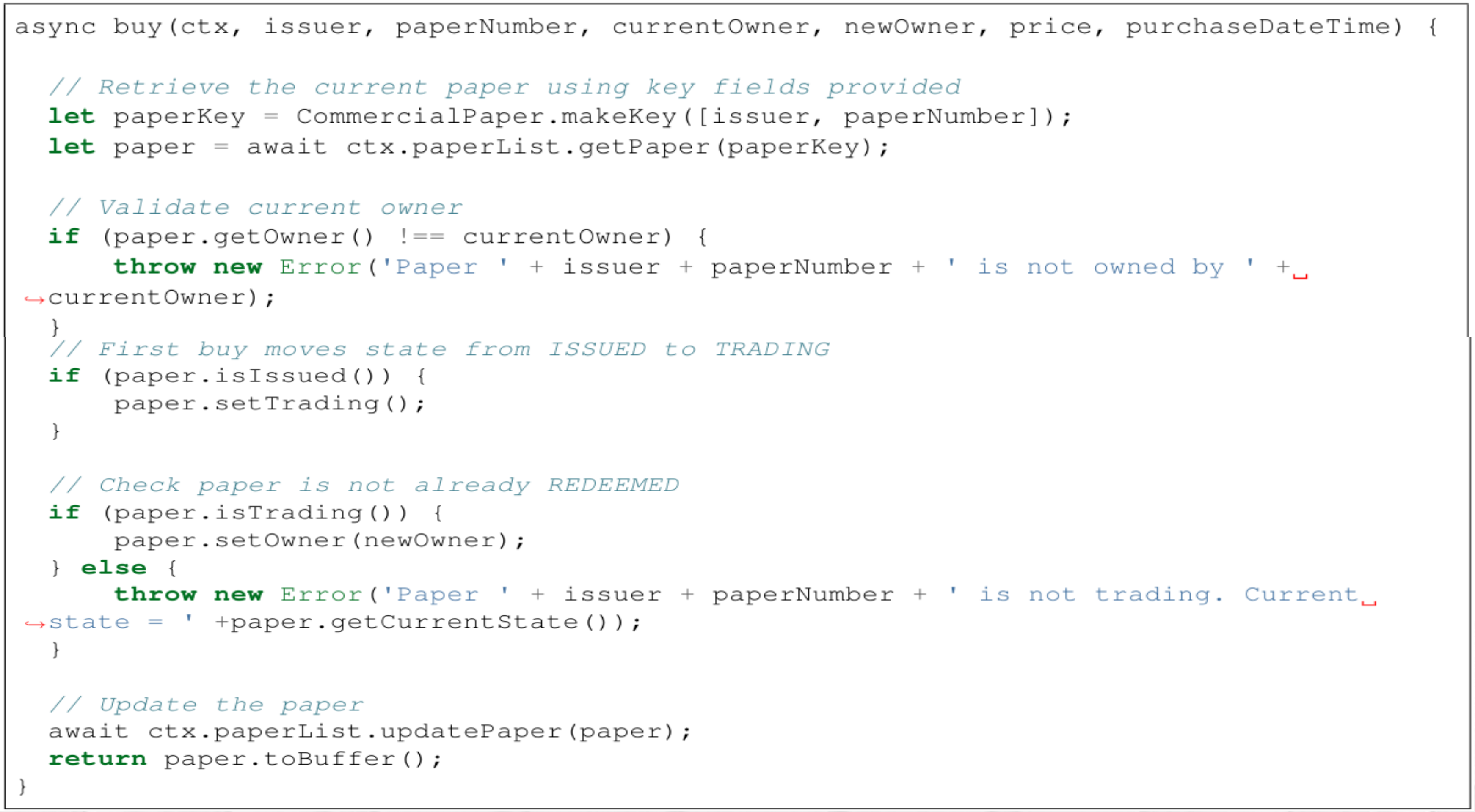
issue method



```
async issue(ctx, issuer, paperNumber, issueDateTime, maturityDateTime, faceValue) {

   // create an instance of the paper
  let paper = CommercialPaper.createInstance(issuer, paperNumber, issueDateTime,␣
↪maturityDateTime, faceValue);

  // Smart contract, rather than paper, moves paper into ISSUED state
  paper.setIssued();

  // Newly issued paper is owned by the issuer
  paper.setOwner(issuer);

  // Add the paper to the list of all similar commercial papers in the ledger world␣
↪state
  await ctx.paperList.addPaper(paper);

  // Must return a serialized paper to caller of smart contract
  return paper.toBuffer();
}
```

Buy 트랜잭션

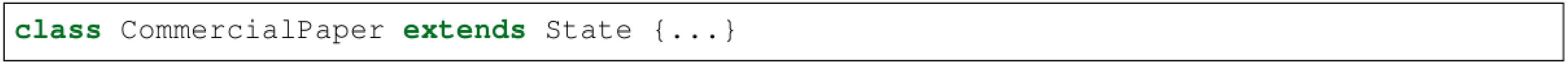
buy transaction



```
async buy(ctx, issuer, paperNumber, currentOwner, newOwner, price, purchaseDateTime) {

  // Retrieve the current paper using key fields provided
  let paperKey = CommercialPaper.makeKey([issuer, paperNumber]);
  let paper = await ctx.paperList.getPaper(paperKey);

  // Validate current owner
  if (paper.getOwner() !== currentOwner) {
      throw new Error('Paper ' + issuer + paperNumber + ' is not owned by ' +␣
↪currentOwner);
  }
  // First buy moves state from ISSUED to TRADING
  if (paper.isIssued()) {
      paper.setTrading();
  }

  // Check paper is not already REDEEMED
  if (paper.isTrading()) {
      paper.setOwner(newOwner);
  } else {
      throw new Error('Paper ' + issuer + paperNumber + ' is not trading. Current␣
↪state = ' +paper.getCurrentState());
  }

  // Update the paper
  await ctx.paperList.updatePaper(paper);
  return paper.toBuffer();
}
```

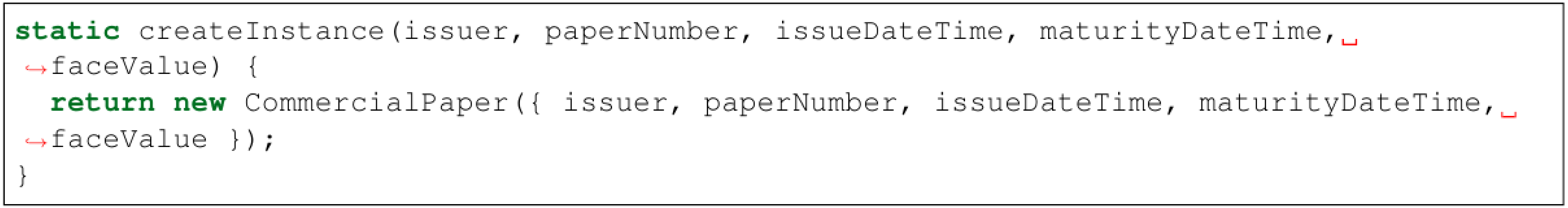
객체의 표현

- Locate the CommercialPaper class in the paper.js file:



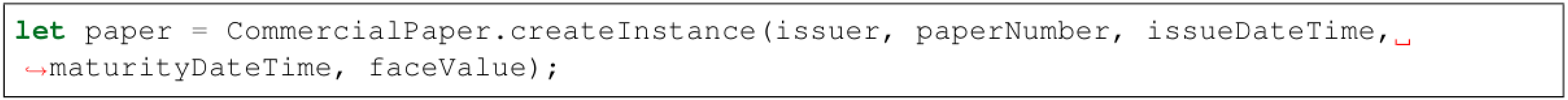
```
class CommercialPaper extends State {...}
```

- initializes a new commercial paper with the provided parameters:



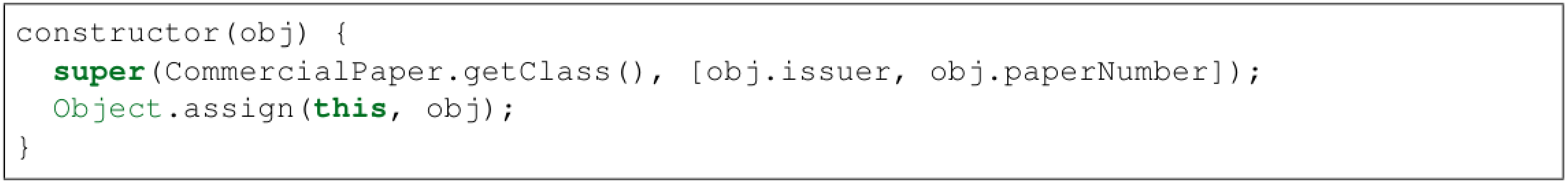
```
static createInstance(issuer, paperNumber, issueDateTime, maturityDateTime,␣
↪faceValue) {
  return new CommercialPaper({ issuer, paperNumber, issueDateTime, maturityDateTime,␣
↪faceValue });
}
```

- this class was used by the issue transaction:



```
let paper = CommercialPaper.createInstance(issuer, paperNumber, issueDateTime,␣
↪maturityDateTime, faceValue);
```

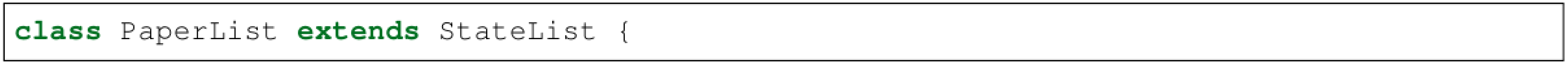
- key is formed from a combination of issuer and paperNumber.



```
constructor(obj) {
  super(CommercialPaper.getClass(), [obj.issuer, obj.paperNumber]);
  Object.assign(this, obj);
}
```

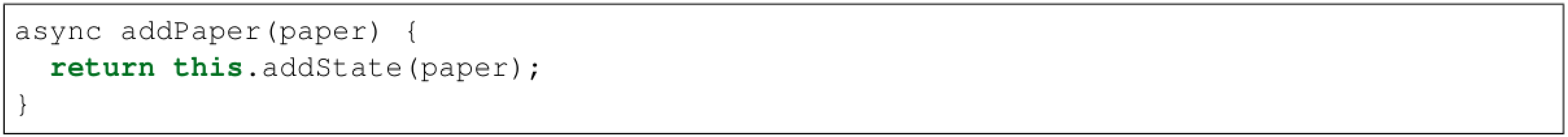
장부에 접근

- locate the PaperList class in the paperlist.js file:



```
class PaperList extends StateList {
```

- The addPaper() method is a simple veneer over the StateList.addState() method:

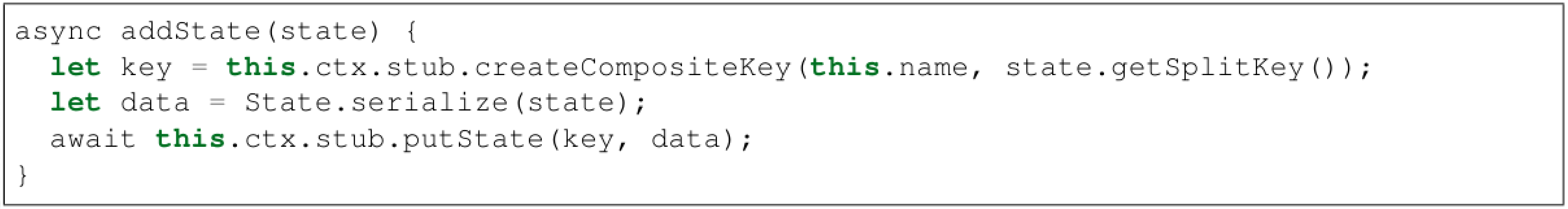


```
async addPaper(paper) {
  return this.addState(paper);
}
```

- StateList class uses the Fabric API putState() to write the commercial paper as state data in the ledger:
- state data in a ledger requires these two fundamental elements:

Key: key is formed with createCompositeKey().

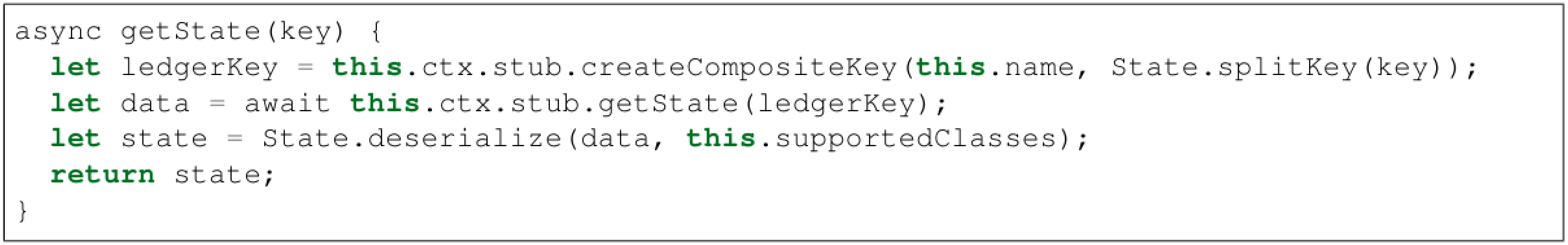
Data: data is simply the serialized form of the commercial paper state.



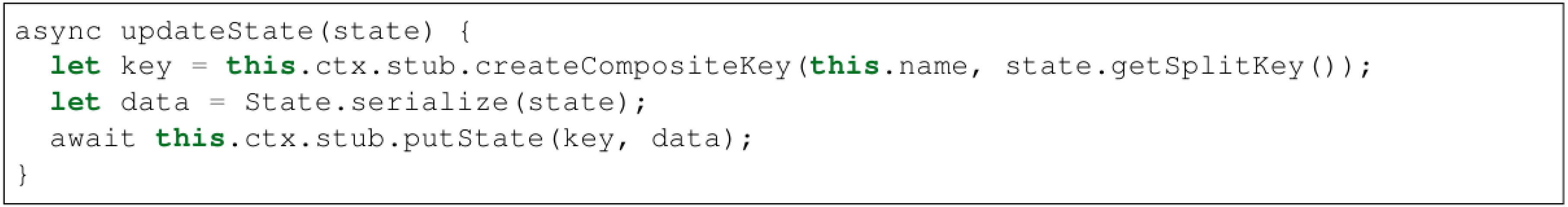
```
async addState(state) {
  let key = this.ctx.stub.createCompositeKey(this.name, state.getSplitKey());
  let data = State.serialize(state);
  await this.ctx.stub.putState(key, data);
}
```

장부에 접근 2

- The StateList getState() and updateState() methods work in similar ways:



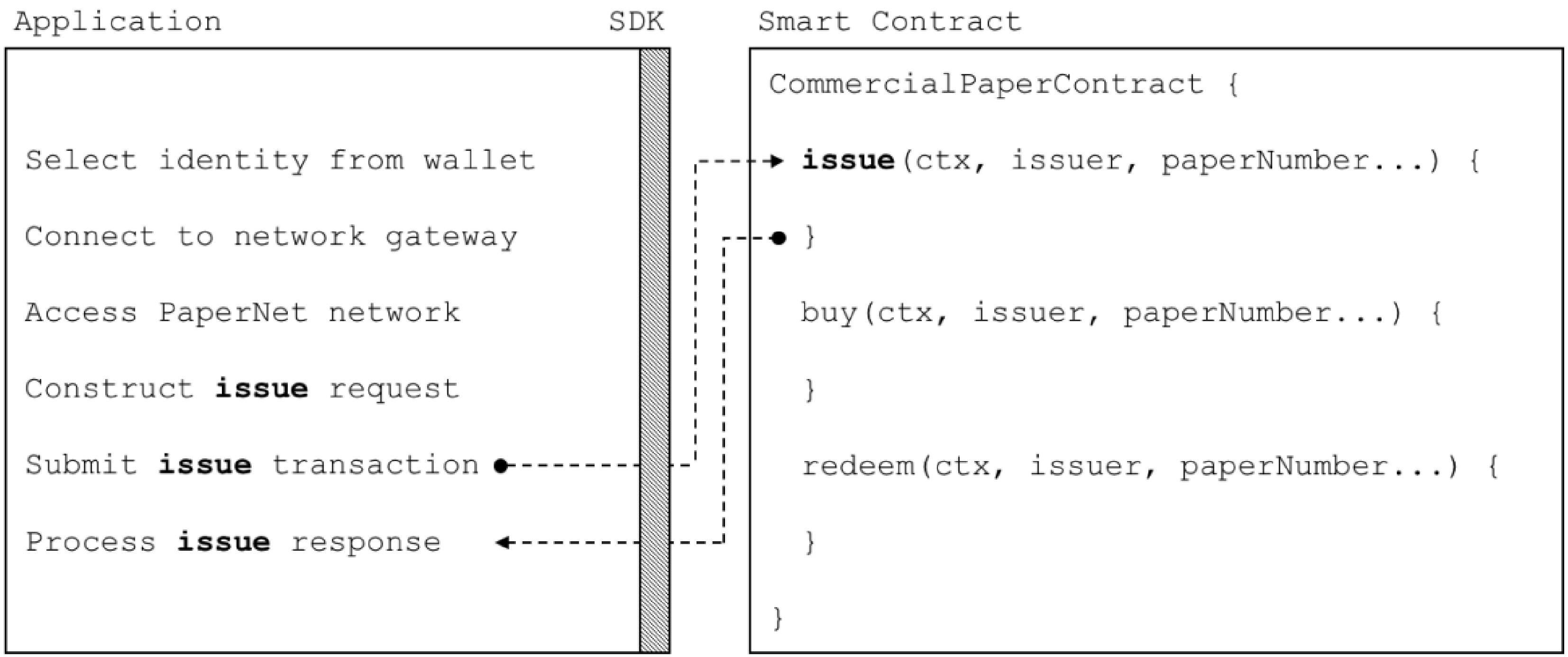
```
async getState(key) {
  let ledgerKey = this.ctx.stub.createCompositeKey(this.name, State.splitKey(key));
  let data = await this.ctx.stub.getState(ledgerKey);
  let state = State.deserialize(data, this.supportedClasses);
  return state;
}
```



```
async updateState(state) {
  let key = this.ctx.stub.createCompositeKey(this.name, state.getSplitKey());
  let data = State.serialize(state);
  await this.ctx.stub.putState(key, data);
}
```

애플리케이션

Basic Flow

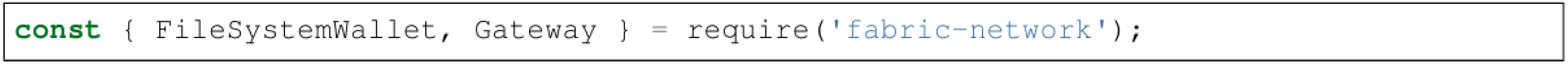


Application
SDK
Smart Contract
Select identity from wallet
Connect to network gateway
Access PaperNet network
Construct **issue** request
Submit **issue** transaction
Process **issue** response
CommercialPaperContract {
issue(ctx, issuer, paperNumber...) {
}
buy(ctx, issuer, paperNumber...) {
}
redeem(ctx, issuer, paperNumber...) {
}
}

PaperNet 애플리케이션은 발행 계약을 요청하기 위하여 어음 스마트컨트랙트를 호출한다.

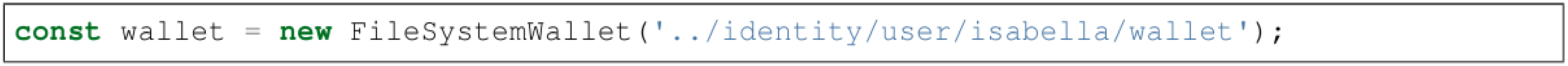
월렛

- Towards the top of issue.js, you'll see two Fabric classes are brought into scope:



```
const { FileSystemWallet, Gateway } = require('fabric-network');
```

- The application uses the Fabric Wallet class as follows:



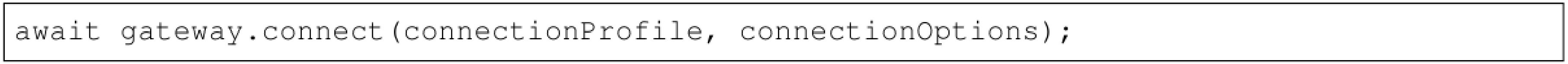
```
const wallet = new FileSystemWallet('../identity/user/isabella/wallet');
```

- The wallet holds a set of identities - X.509 digital certificates - which can be used to access PaperNet or any other Fabric network.
- A wallet holding the digital equivalents of your government ID, driving license or ATM card.

150

게이트웨이

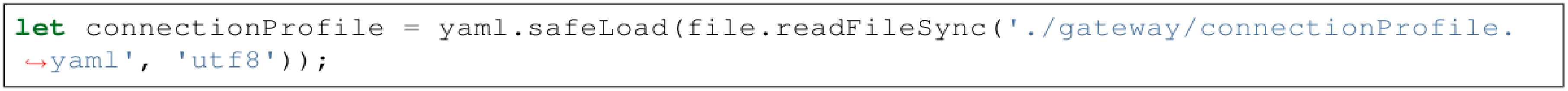
- A gateway identifies one or more peers that provide access to a network - in our case, PaperNet.
- issue.js connects to its gateway:
- gateway.connect() has two important parameters:
  connectionProfile: identifies a set of peers as a gateway to PaperNet
  connectionOptions: a set of options used to control how issue.js interacts with PaperNet



```
await gateway.connect(connectionProfile, connectionOptions);
```

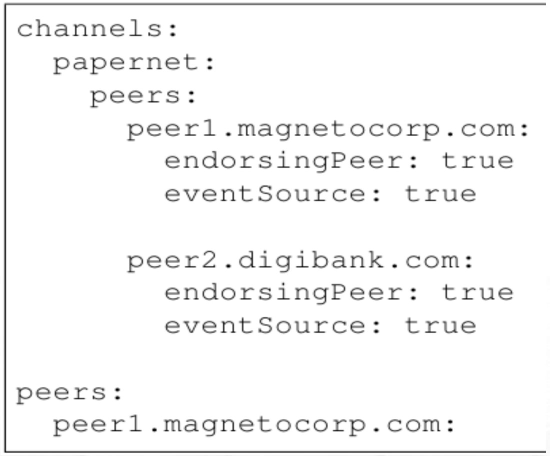
연결정보

- Connection profile (./gateway/connectionProfile.yaml) uses YAML, making it easy to read.
- It was loaded and converted into a JSON object:

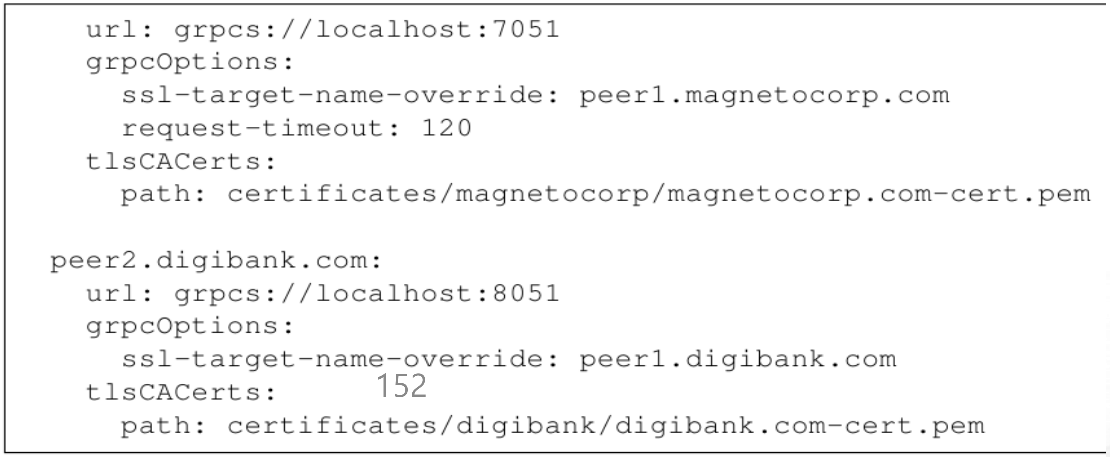


```
let connectionProfile = yaml.safeLoad(file.readFileSync('./gateway/connectionProfile.
↪yaml', 'utf8'));
```

- 정보보유 (Peers, network channels, network orderers, organizations, CA)



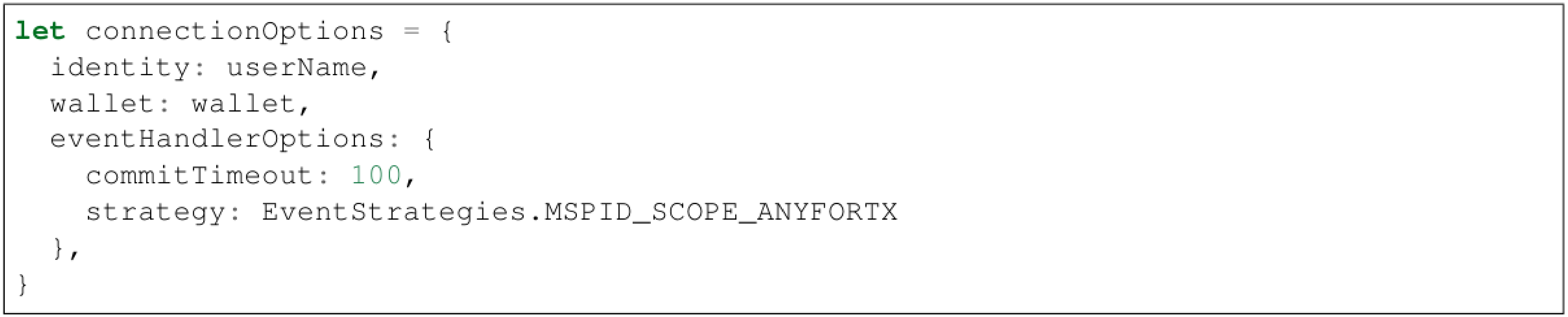
```
channels:
  papernet:
    peers:
      peer1.magnetocorp.com:
        endorsingPeer: true
        eventSource: true

      peer2.digibank.com:
        endorsingPeer: true
        eventSource: true

peers:
  peer1.magnetocorp.com:
```



```
    url: grpcs://localhost:7051
    grpcOptions:
      ssl-target-name-override: peer1.magnetocorp.com
      request-timeout: 120
    tlsCACerts:
      path: certificates/magnetocorp/magnetocorp.com-cert.pem

  peer2.digibank.com:
    url: grpcs://localhost:8051
    grpcOptions:
      ssl-target-name-override: peer1.digibank.com
    tlsCACerts:      152
      path: certificates/digibank/digibank.com-cert.pem
```

게이트웨이 2

- connectionOptions object:



```
let connectionOptions = {
  identity: userName,
  wallet: wallet,
  eventHandlerOptions: {
    commitTimeout: 100,
    strategy: EventStrategies.MSPID_SCOPE_ANYFORTX
  },
}
```

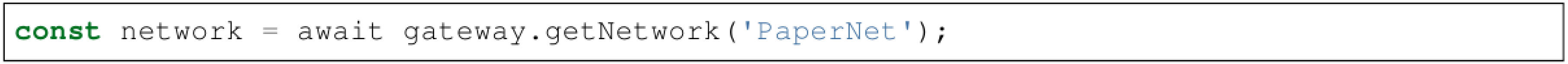
- EventStrategies.MSPID_SCOPE_ANYFORTX : SDK can notify an application after a single MagnetoCorp peer has confirmed the transaction
- EventStrategies.NETWORK_SCOPE_ALLFORTX: requires that all peers from MagnetoCorp and DigiBank to confirm the transaction.

153

채널

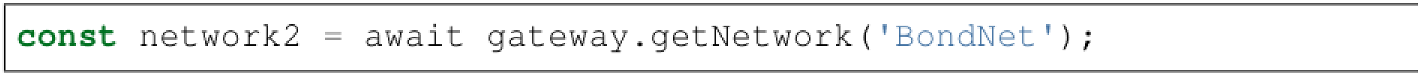
Network channel

- Application selects a particular channel:



```
const network = await gateway.getNetwork('PaperNet');
```

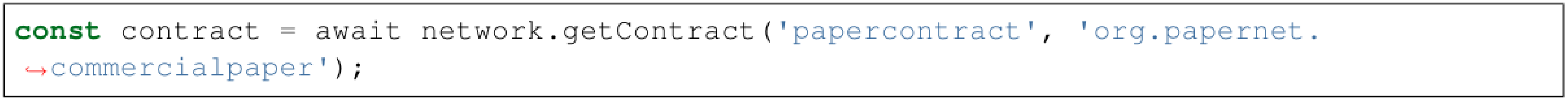
- If the application wanted to access another network, BondNet, at the same time:



```
const network2 = await gateway.getNetwork('BondNet');
```

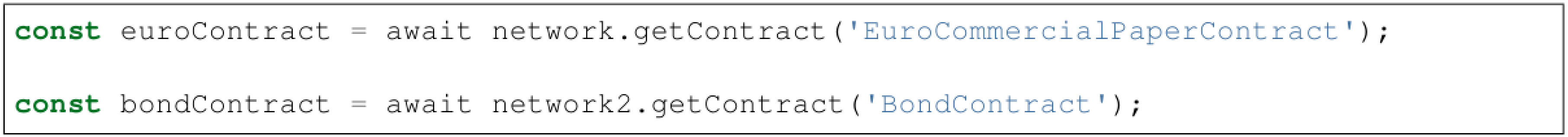
Construct request

- To issue a commercial paper needs to use CommercialPaperContract. To access this smart contract:



```
const contract = await network.getContract('papercontract', 'org.papernet.
↪commercialpaper');
```

- If our application simultaneously required access to another contract in PaperNet or BondNet:

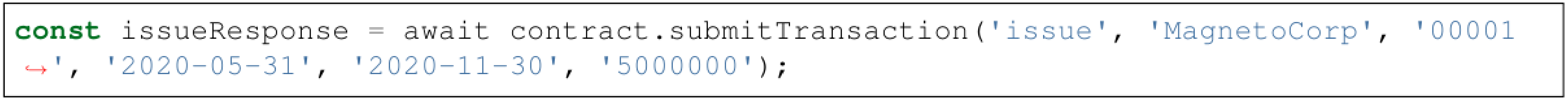


```
const euroContract = await network.getContract('EuroCommercialPaperContract');

const bondContract = await network2.getContract('BondContract');
```

154

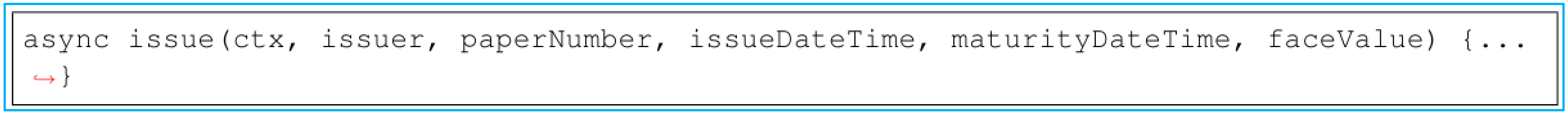
트랜잭션 제출

- Submitting a transaction is a single method call to the SDK:



```
const issueResponse = await contract.submitTransaction('issue', 'MagnetoCorp', '00001
↪', '2020-05-31', '2020-11-30', '5000000');
```

- submitTransaction() parameters will be passed to the issue() method in the smart contract,
- and used to create a new commercial paper. Recall its signature:

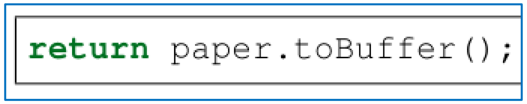


```
async issue(ctx, issuer, paperNumber, issueDateTime, maturityDateTime, faceValue) {...
↪}
```

155

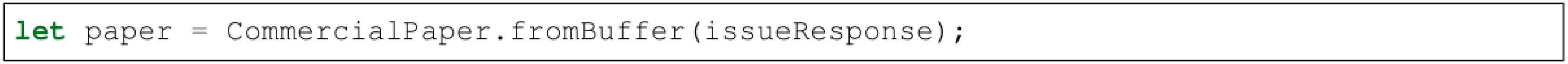
트랜젝션 응답

- issue transaction returns a commercial paper response:



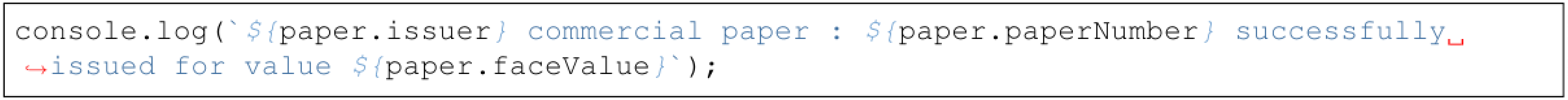
```
return paper.toBuffer();
```

- class method CommercialPaper.fromBuffer() to rehydrate the response buffer as a commercial paper:



```
let paper = CommercialPaper.fromBuffer(issueResponse);
```

- This allows paper to be used in a natural way in a descriptive completion message:



```
console.log(`${paper.issuer} commercial paper : ${paper.paperNumber} successfully␣
↪issued for value ${paper.faceValue}`);
```

전체프로세스 다시보기

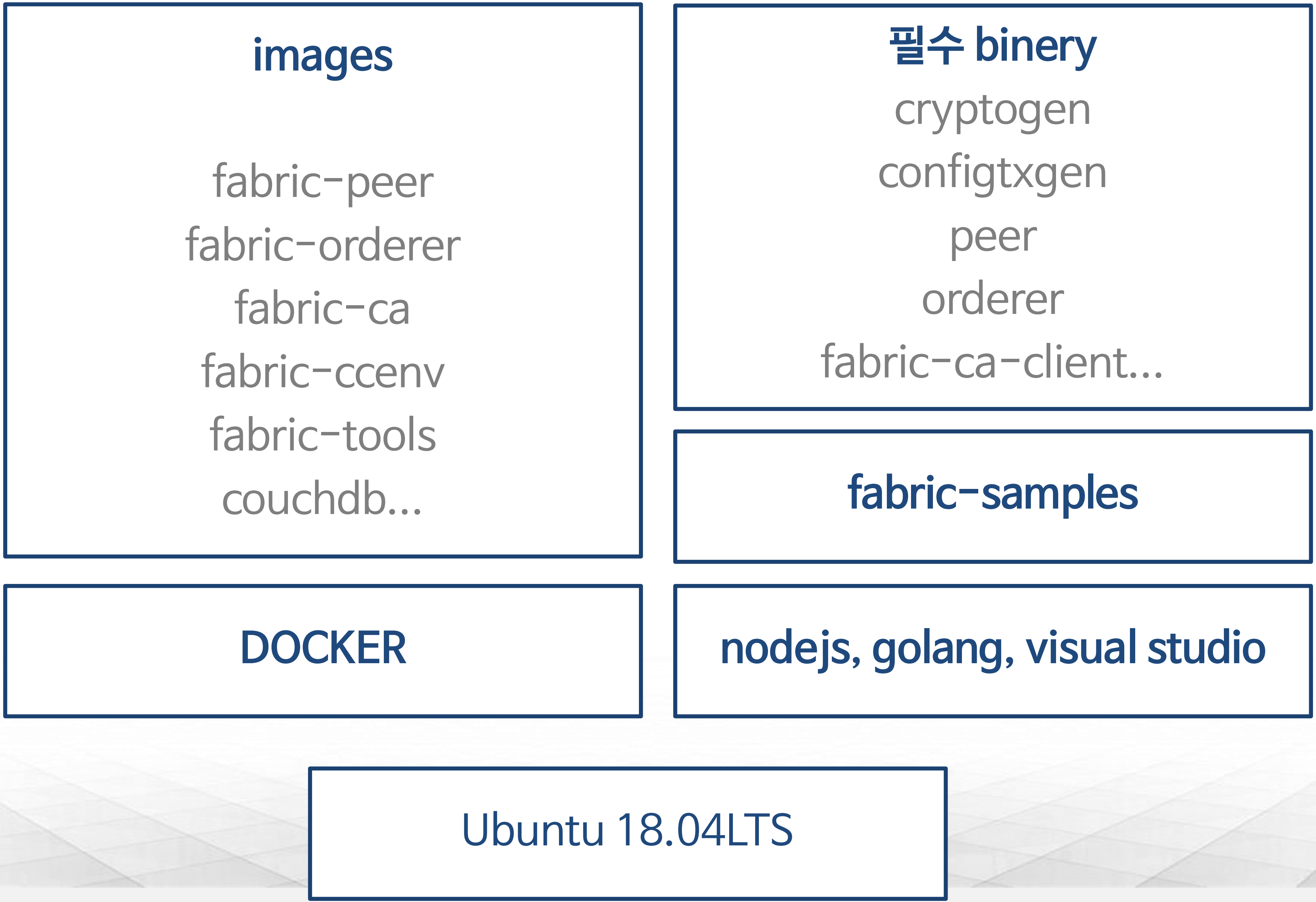
기획
기업어음
비지니스네트워크
비지니스프로세스
비지니스모델

환경설정
1. 네트워크
org1 (MC) peer+ca+db
org2 (DB) peer+ca+db
orderer (solo or raft)
채널

2. 체인코드
papercontract
issue, buy, redeem
go java nodejs
설치, 배포, 보증정책?

3. 웹서버 연동
nodejs java go python
fabric-ca-client
fabric-network
connection.json
인증서

패브릭 설치



images

fabric-peer
fabric-orderer
fabric-ca
fabric-ccenv
fabric-tools
couchdb...

필수 binery

cryptogen
configtxgen
peer
orderer
fabric-ca-client...

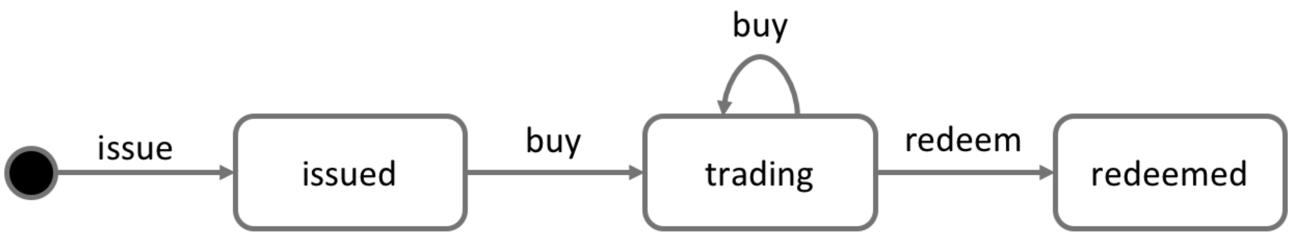
fabric-samples

DOCKER

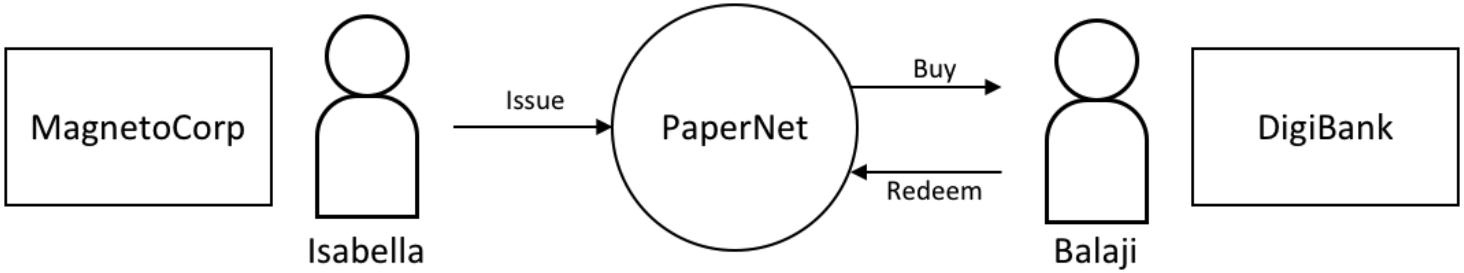
nodejs, golang, visual studio

Ubuntu 18.04LTS

기획

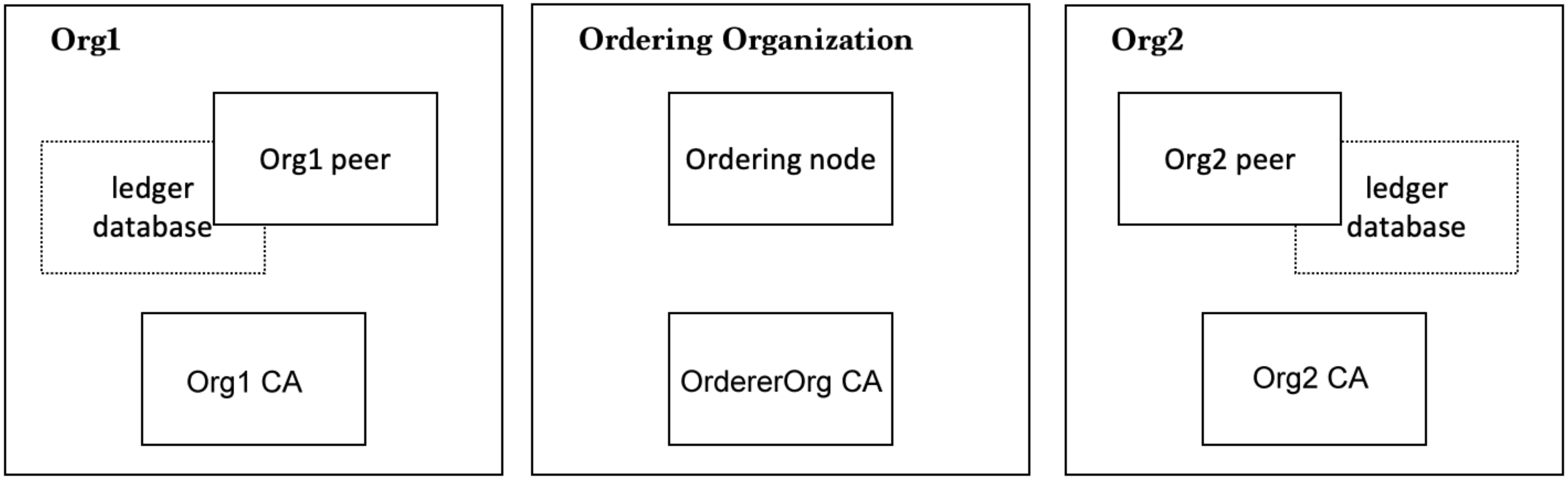


```
Issuer: MagnetoCorp
Paper: 00001
Owner: DigiBank
Issue date: 31 May 2020
Maturity date: 30 Nov 2020
Face value: 5M USD
Current state: trading
```

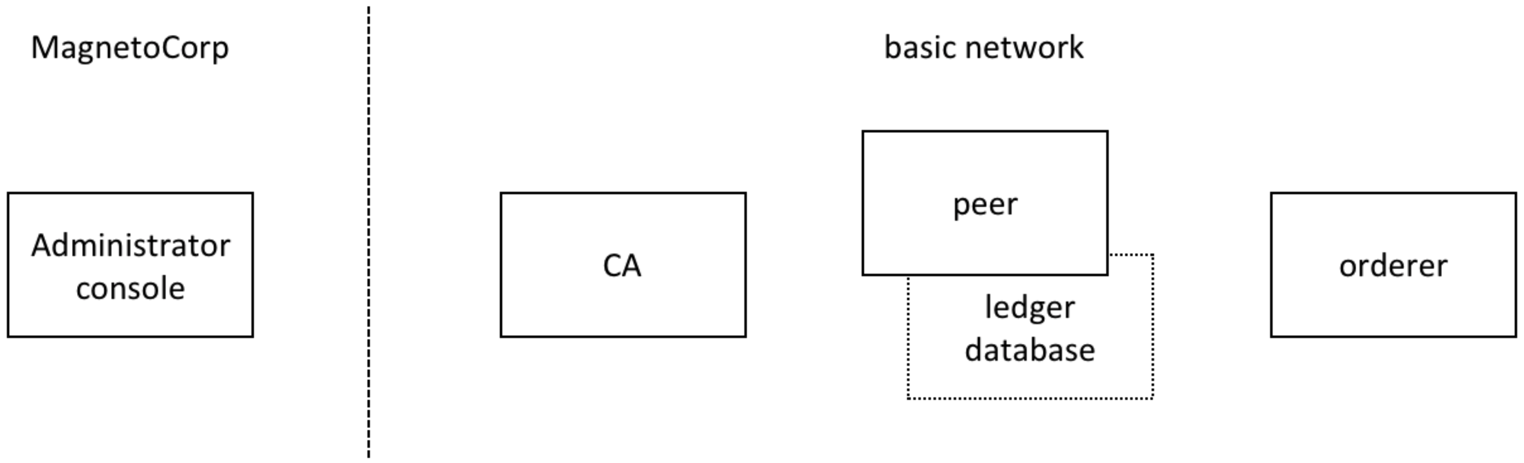


네트워크 구성

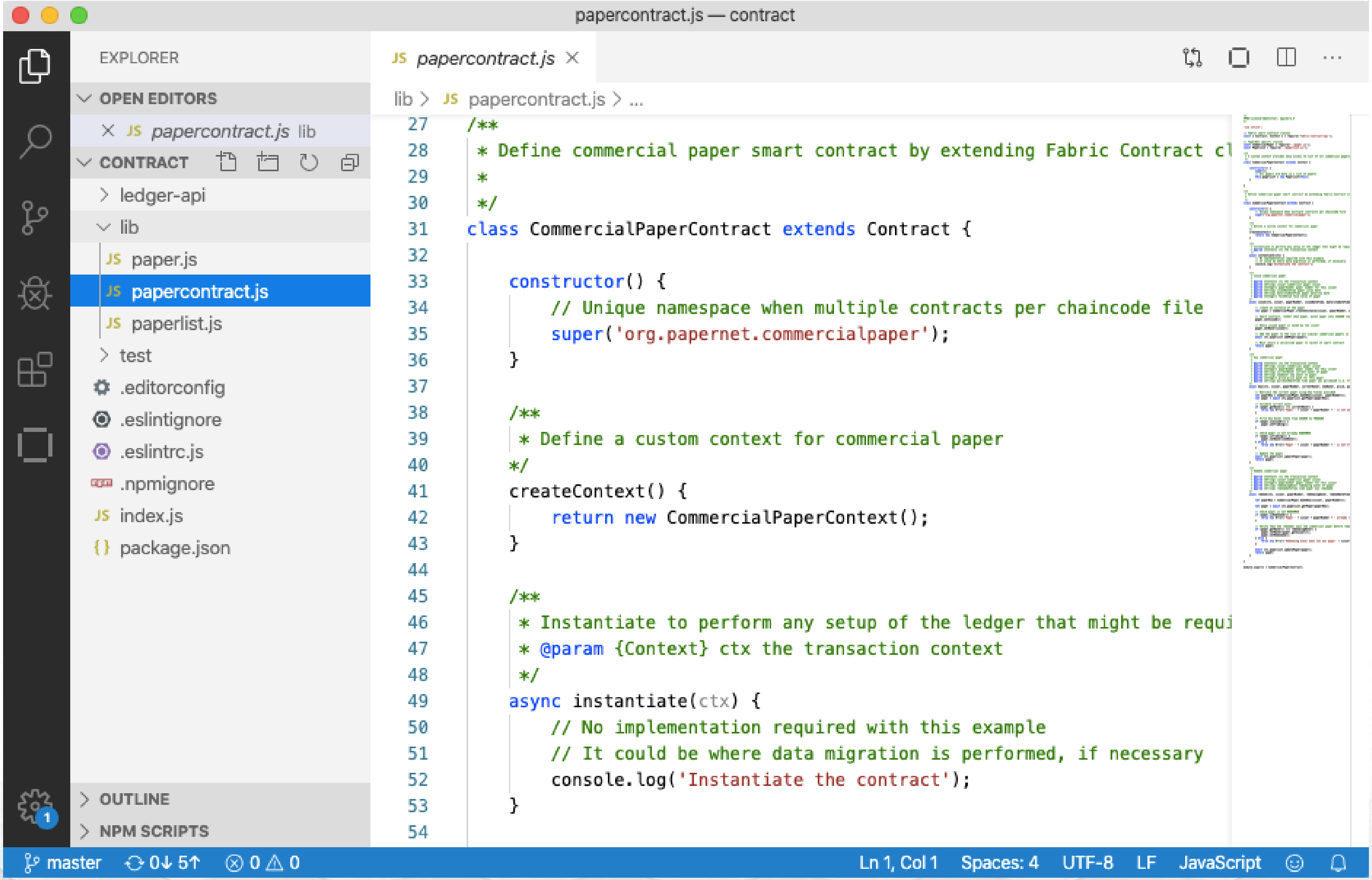
Fabric test network



네트워크 READY

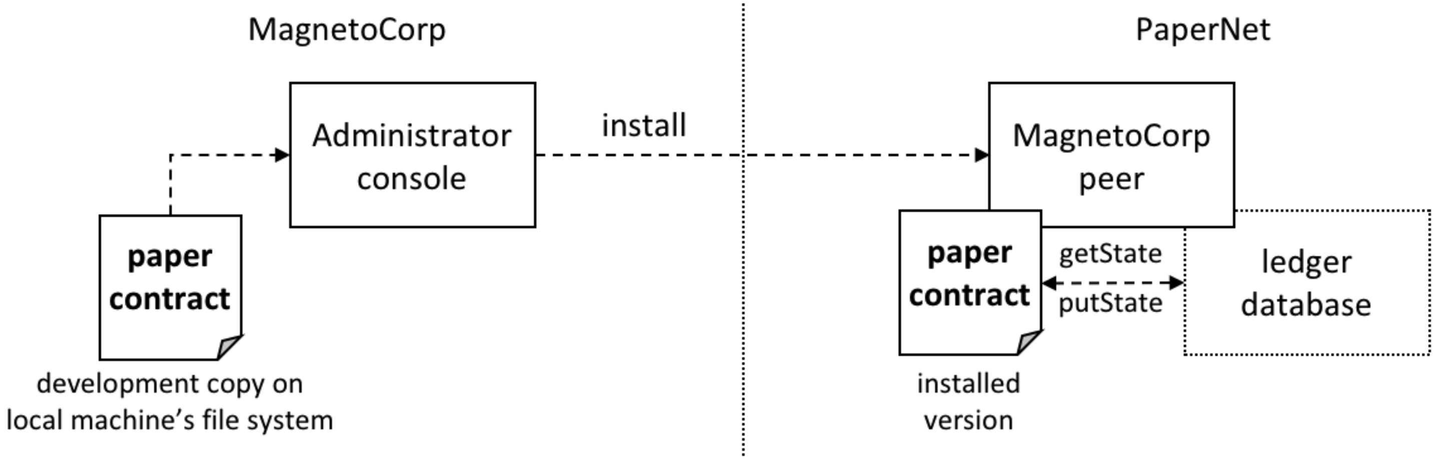

MagnetoCorp
basic network
Administrator console
CA
peer
ledger database
orderer

체인코드 개발

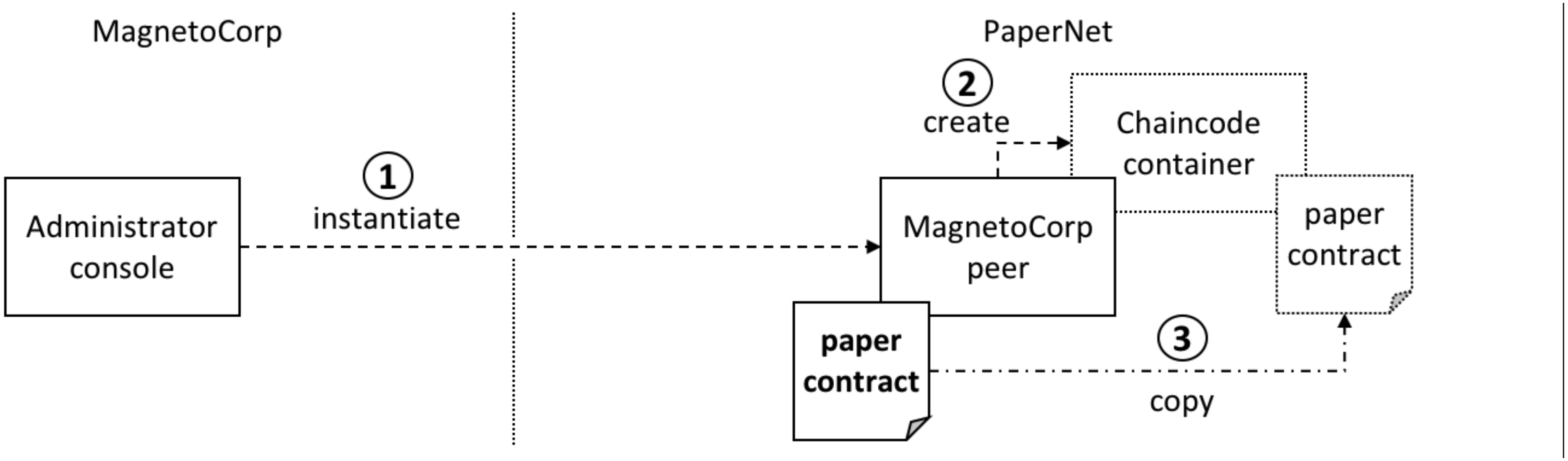


```javascript
/**
 * Define commercial paper smart contract by extending Fabric Contract cl
 *
 */
class CommercialPaperContract extends Contract {

    constructor() {
        // Unique namespace when multiple contracts per chaincode file
        super('org.papernet.commercialpaper');
    }

    /**
     * Define a custom context for commercial paper
    */
    createContext() {
        return new CommercialPaperContext();
    }

    /**
     * Instantiate to perform any setup of the ledger that might be requi
     * @param {Context} ctx the transaction context
     */
    async instantiate(ctx) {
        // No implementation required with this example
        // It could be where data migration is performed, if necessary
        console.log('Instantiate the contract');
    }

```

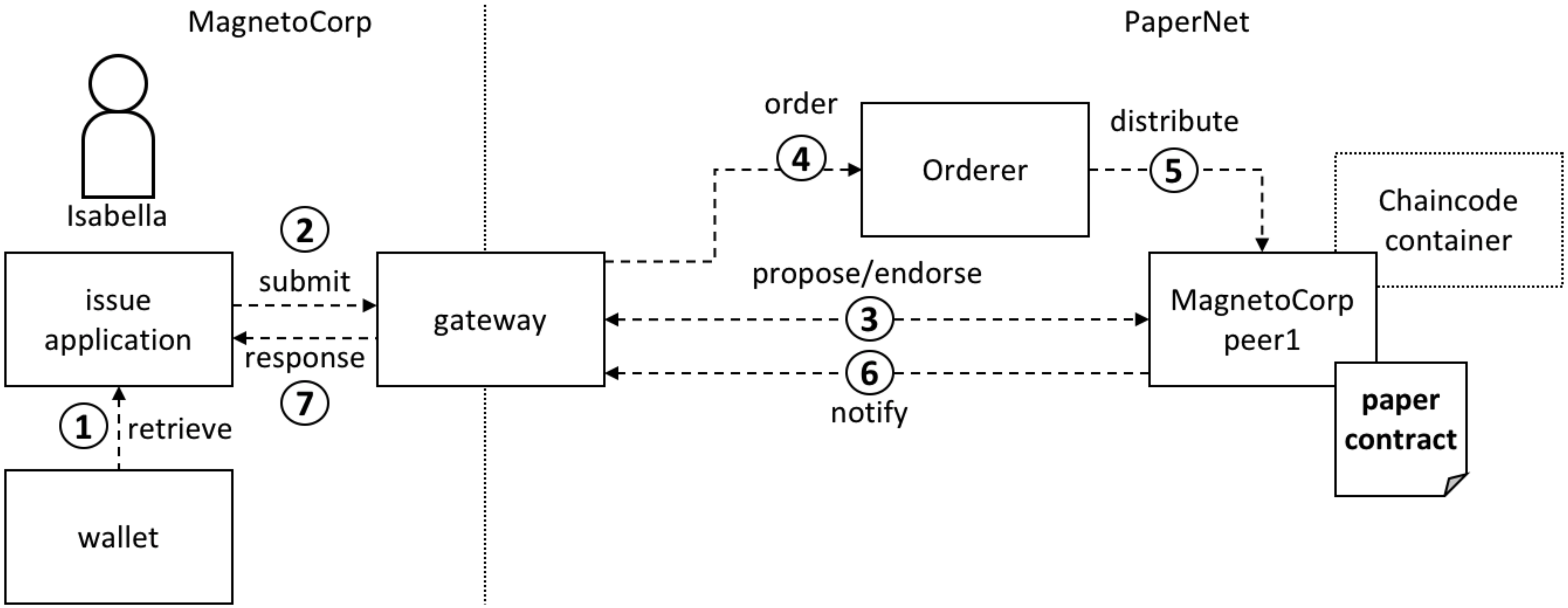
체인코드 설치


MagnetoCorp
PaperNet
Administrator console
install
MagnetoCorp peer
paper contract
development copy on local machine's file system
paper contract
getState
putState
ledger database
installed version

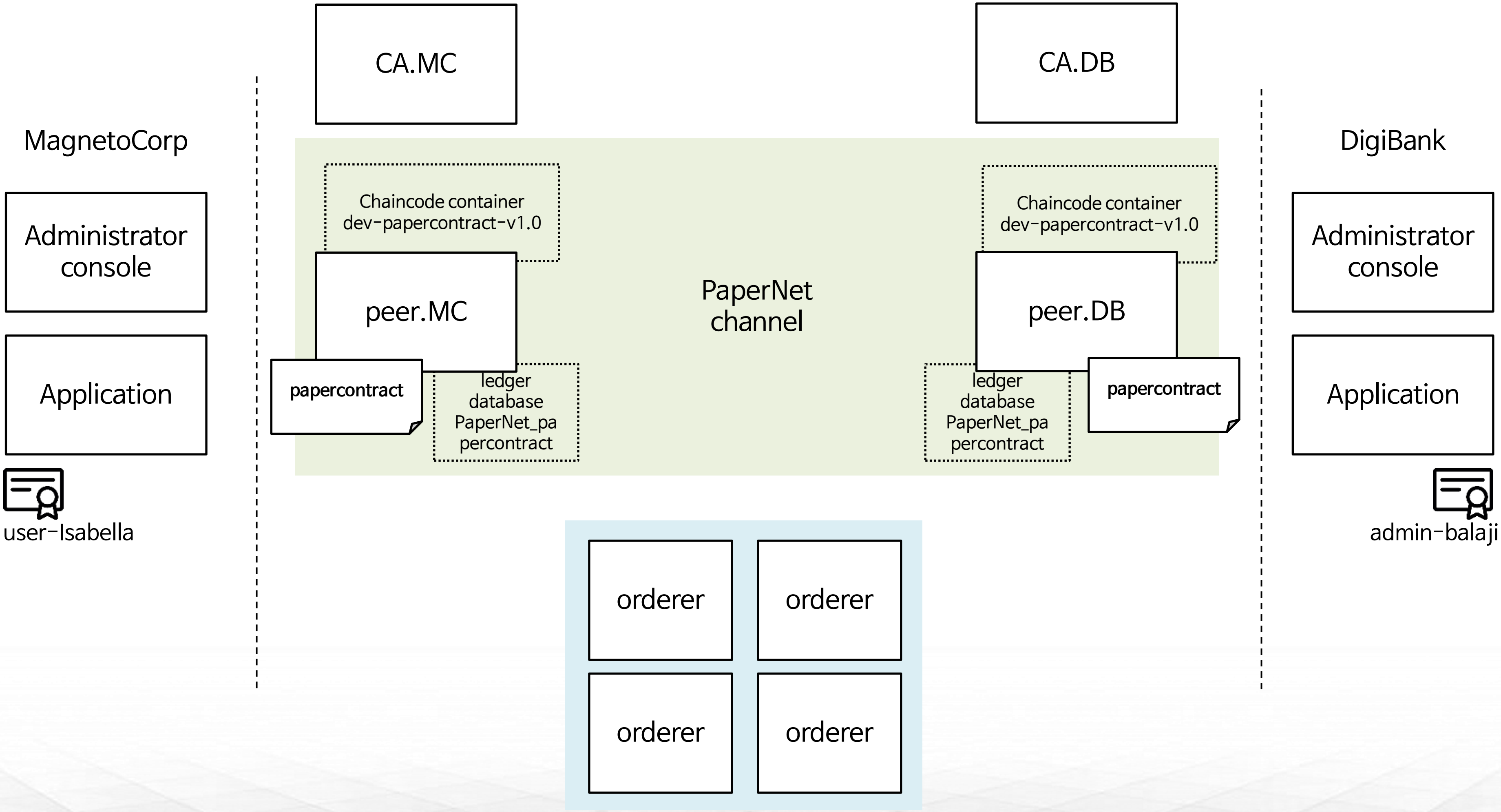
체인코드 배포


MagnetoCorp
PaperNet
① instantiate
Administrator console
② create
Chaincode container
MagnetoCorp peer
paper contract
paper contract
③
copy

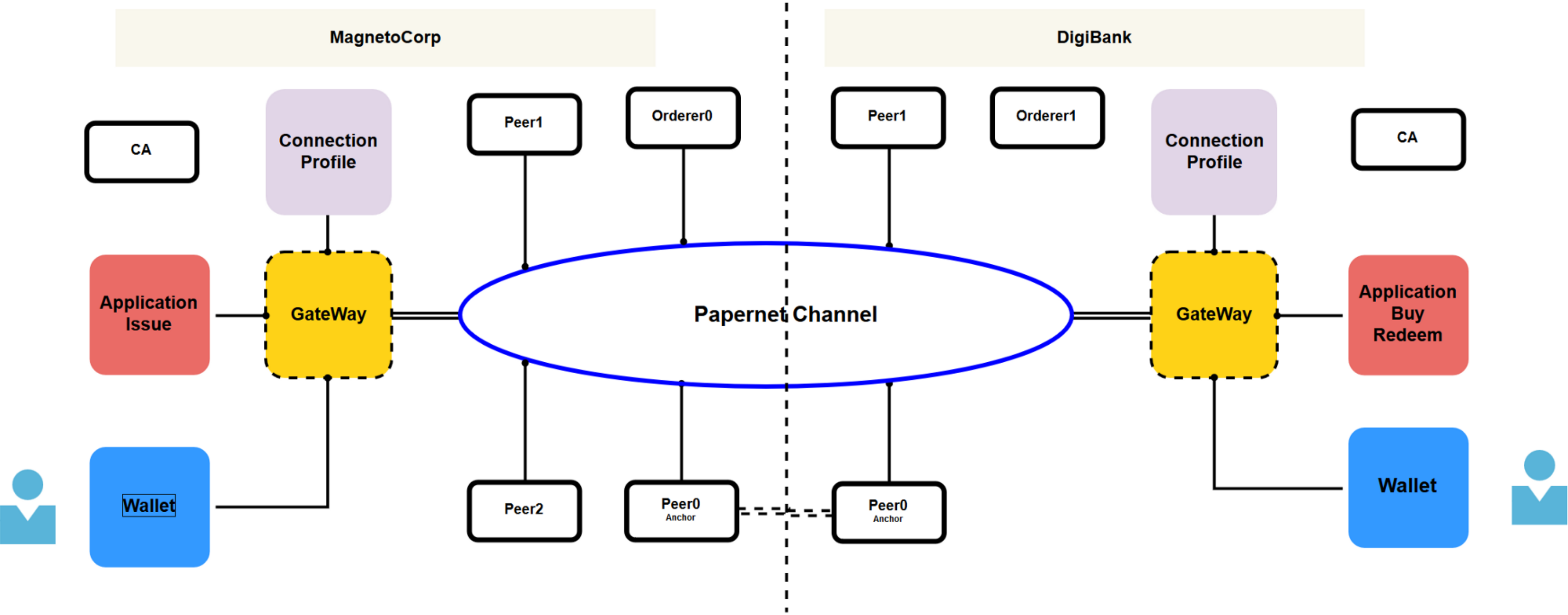
어플리케이션 구동 – MC


MagnetoCorp
PaperNet
Isabella
issue application
wallet
1 retrieve
2 submit
response 7
gateway
order 4
Orderer
distribute 5
Chaincode container
propose/endorse 3
MagnetoCorp peer1
6 notify
paper contract

어플리케이션 구동 - DB


CA.MC
CA.DB
MagnetoCorp
DigiBank
Administrator console
Application
user-Isabella
Chaincode container dev-papercontract-v1.0
peer.MC
papercontract
ledger database PaperNet_papercontract
PaperNet channel
Chaincode container dev-papercontract-v1.0
peer.DB
ledger database PaperNet_papercontract
papercontract
Administrator console
Application
admin-balaji
orderer
orderer
orderer
orderer

papernet 구성 예


MagnetoCorp
DigiBank
CA
Connection Profile
Peer1
Orderer0
Application Issue
GateWay
Papernet Channel
Wallet
Peer2
Peer0 Anchor
Peer1
Orderer1
Connection Profile
CA
GateWay
Application Buy Redeem
Peer0 Anchor
Wallet