CREDO – Credentialing Module

Professionals Credentials through Verifiable Claims implemented in Blockchain Technology

Version 1.5.

25th of November 2018

Contents

1. Project Information

1.1. Project Title

CREDO - Credentialing Module. Professionals Credentials through Verifiable Claims implemented with Hyperledger Fabric.

1.2. Project Scope and Duration

The participants will create a project, called Capstone Project, for the CFDev Course @ York University. The project will need to be finalized in the following

- Start Date: 15th of November
- End Date: 30th of November
  Duration: 2 weeks.

1.3. Project Participants & IP

Project Participants:

- Osman B
- Ricardo A
- Radu V

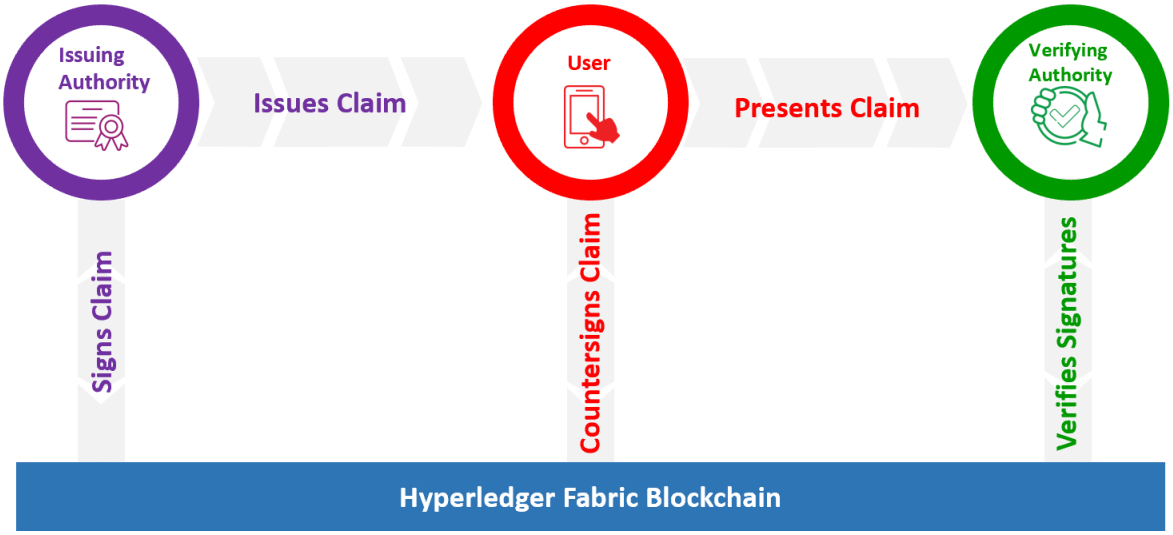
It is herewith agreed upon that all the IP resulted in this project is a shared equally by the participants, and it can be used by any of the participants at their own will, with no liabilities or encumberments whatsoever.

1.4. Project Description

Project will have 3 actors: users who will co-sign credentials, and query their own credentials
Issuing authorities who will issue the credentials, and co-sign them with the user in the ledger
Verifying authorities which will verify the credentials (DID's) of the user, with the permission of the user, from the ledger.

The credentials are written and queried via the chaincode which we will write as part of the project.

We intent to build a project where **credentials**, can be issued by an organisation, the user will sign it, together with the organisation – called Issuing Authority, and will be placed in a ledger as DID's (decentralized ID's)



1.5. Project Architecture

1.5.1. Technical Architecture

We will install the Hyperledger Fabric Nodes as per the instructions provided in the course.

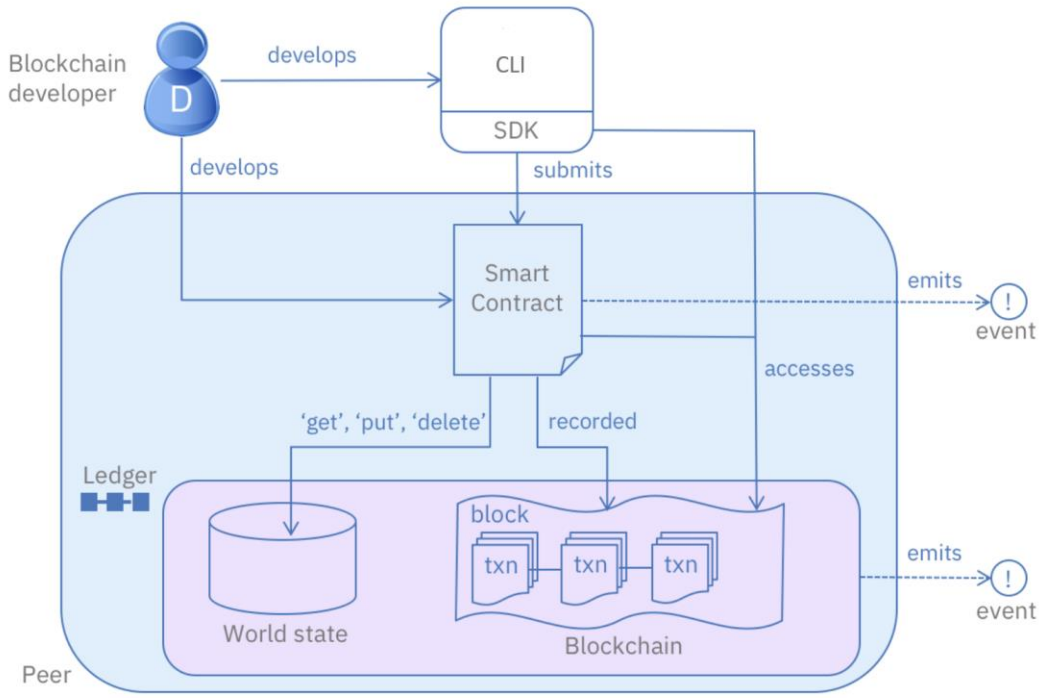
1. Ubuntu AWS image (for environments)
2. Node / NPM (for Fabric code)
3. Go Lang (for Fabric)
4. Docker (for Nodes)

Technologies we intent to write for the scope of the project:

- Node JS for the chaincode (smart contract)

1.5.2. Application Architecture

The following diagram depicts the way our small application, CREDO App, will interact with the smart contract (chaincode) and the ledger, via the CLI.



1.5.2.1. Chaincode (Smart Contract):

In order to define the actual structure of the chaincode, and the code itself, we're defining the variables, claims statuses, and the actual functions / methods to implement the chaincode itself.

Variables:

UserID = number
ClaimType = char
IssuingAuth = number
VerifyingAuth = number
Value = numeric
ExpDate = date

Claims **status**:

- **Submitted** (by the organisation i/e Org1). The claim will be pending signature from user.
- **Signed** (by the user). The claim will be active.

Chaincode Functions / Methods:

UserRecord

- **appendClaim** <userID> <claimType> <claimValue> <submitOrganization> <expirationDate>
- **signClaim** <userID> <claimType>
- **verifyClaim** <userID> <claimType>
- **queryAllClaims**

Example of **verifyClaim** method (in **Credo.js**):

```
async verifyClaim(stub, args){
  console.info('============= START : Verify Claim ===========');

  if (args.length != 2) {
    throw new Error('Incorrect number of arguments. Expecting 2');
  }
  let claimID = args[0]+ "/" + args[1];
  console.log('claimID:' + claimID);
  let claimBytes = await stub.getState(claimID);
  console.log('claimBytes:' + claimBytes);
  let claim = JSON.parse(claimBytes);

  var verifyStatus = {
    userID: args[0],
    type: parseInt(args[1]),
    verified: (claim.status == ClaimStatus.ACTIVE),
  };
```

1.5.2.2. Ledger Structure:

- **UserID,**
- Issuing Authority,
- Verifying Authority,
- **ClaimType,**
- Value,
- ExpDate,
- **ClaimStatus**

1.6. Description of Activities – User Stories

We will develop user stories to execute the project.

Below are the major user stories, details and participants will be added:

1. As a dev-ops: to install the ledger, world state, and node peer, orderer, node, certifying authorities (CA's) – certificates, etc.
2. As a dev-ops: ensuring SOLO setup, MSP functional.
3. **As an IssuingAuthority: to append and sign in the ledger**
4. **As a User: to sign own claim**
5. **As a VerifyingAuthority: to view from the ledger**
6. **As a User: to view own claims**

2. Project Outcomes and Findings

Expected deliverables / outcomes:

1. **Install** the **Hyperledger Fabric**
2. **Simulate multiple organisations** and **users**
3. **Manage claims**
    3.1 To **append** a new claim
    3.2 To **sign** a claim
    3.3 To **verify** claim
    3.4 To **query all** claims

We successfully completed all items mentioned above, including the management of the claims (items 3.1., 3.2., 3.3. and 3.4.)

The following output, for exemplification purposes only, is for 3.4. To query all claims (2 claims inserted).

root@28ae3f027af4:/opt/gopath/src/github.com/hyperledger/fabric/peer# ./scripts/**queryAllClaims.sh**
[{"Key":"user1/1","Record":{"userID":"user1","type":1,"value":"1","issuingAuthority":"General Hospital","expDate":"2020-10-30T00:00:00.000Z","status":1}},{"Key":"user2/2","Record":{"userID":"user2","type":2,"value":"1","issuingAuthority":"TGH","expDate":"2018-12-31T00:00:00.000Z","status":1}}]

The code can be found in GitHub.