

A PROJECT REPORT

ON

Blockchain Support for Tracking Liquid Fuel Distribution

ATHENS INFORMATION TECHNOLOGY, GREECE

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR
THE AWARD OF THE 45 DAYS SUMMER INTERNSHIP PROGRAM
(VISHWANIKETAN-CGC NETWORK UG-FELLOWSHIP 2019)

BY

Anish Katkamwar

Atharva Agade

Shubhankar Gaikwad

UNDER THE SUPERVISION OF

Dr. Sofoklis Efremidis



ATHENS INFORMATION TECHNOLOGY

44 Kifisias Av. Monumental Plaza, Building C

15125 Marousi, Athens, Greece

Acknowledgment

A single person cannot carry out a humble brief that any project works with success. Nevertheless, we have made an attempt to this report to express our deepest gratitude to all those who have contributed to complete this project work either directly or indirectly.

At the very outside we are very thankful to the **Athens Information Technology, Greece and Vishwaniketan-CGC Network, India** for giving us this opportunity to work on this project and making available all kinds of resources in order to complete this project successfully.

We are very much thankful to **Dr. Sofoklis Efremidis** for his motivation, guidance and co-operation in the completion of this 45-days summer training program (Vishwaniketan UG-fellowship), as without his guidance it would have been difficult to overcome the challenges faced during the execution of this project.

Last but not the least we heartfelt thanks to **Prof. Sneha Patil**, our colleagues and our parents who have directly or indirectly guided and helped us in completion of the project and for giving us an unending support right from the stage this idea was conceived. We also acknowledge the research work done by all worldwide researchers in this field.

Date:_____/____/____

Anish Katkamwar

Atharva Agade

Shubhankar Gaikwad

Place: Athens, Greece

Abstract

We have created a blockchain based support system for tracking liquid fuel distribution. The adoption of a blockchain system for payment could speed up the process significantly & cut cost at the same time. Using blockchain, various multiple parties involved in operations from upstream to downstream, such as gasoline producer, shipper, supplier, transporter, regulator & settlement provider are brought on to a single platform for enhancing trust & visibility. Also, by using smart contracts, a majority of the operations are automated and data integrity is maintained throughout the supply chain management of gasoline. At a global scale, data becomes available and more reliable.

Contents

Sr. No.	Topic	Page No.
	Cover Page	i
	Acknowledgement	ii
	Abstract	iii
	Content	Iv
	List of figures	v
01	INTRODUCTION	1
	1.1 Introduction to blockchain technology	
	1.2 Hyperledger Fabric	
	1.3 Hyperledger Composer	
02	PROBLEM STATEMENT	10
03	LITERATURE REVIEW	11
04	SOFTWARE MODEL AND DESIGN	12
	4.1 Components of supply chain	
	4.2 Key Concepts	
	4.3 Architecture	
05	WORKING	16
	5.1 Creating a business network	
	5.2 Deploy to rest api	
	5.3 Angular app and composer playground	
06	CONCLUSIONS	26
07	REFERENCES	27

CHAPTER 01

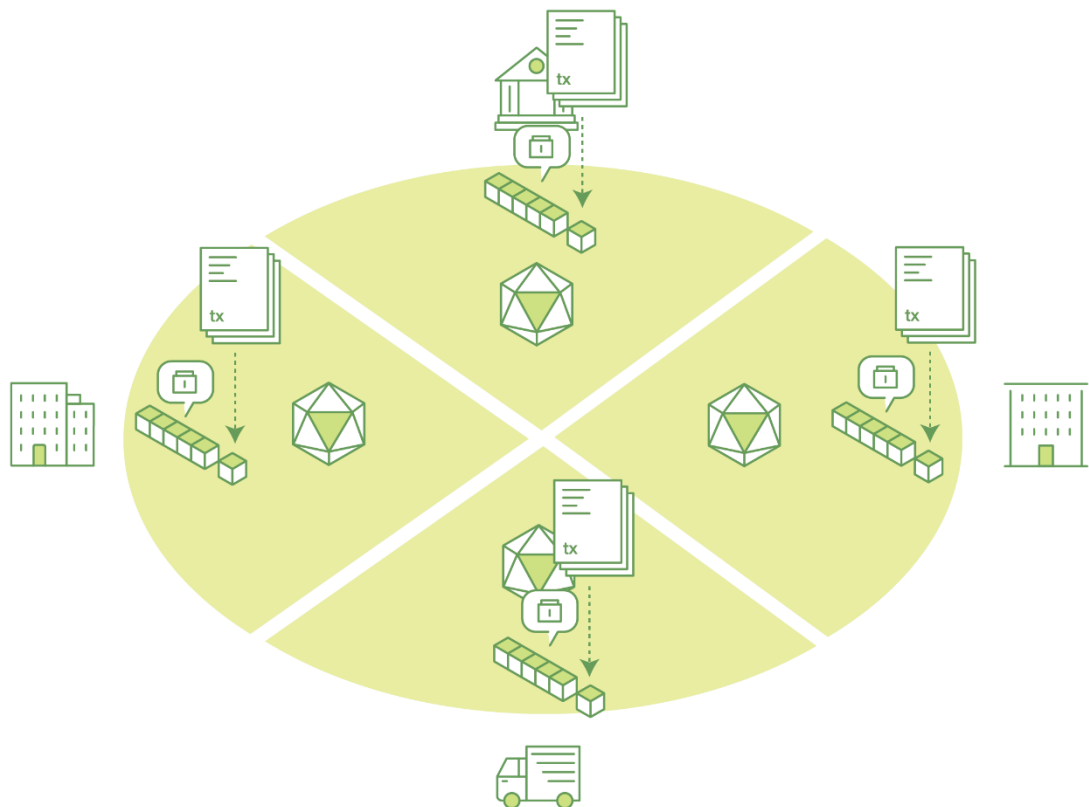
INTRODUCTION

1.1 Introduction to blockchain technology

A Distributed Ledger

At the heart of a blockchain network is a distributed ledger that records all the transactions that take place on the network.

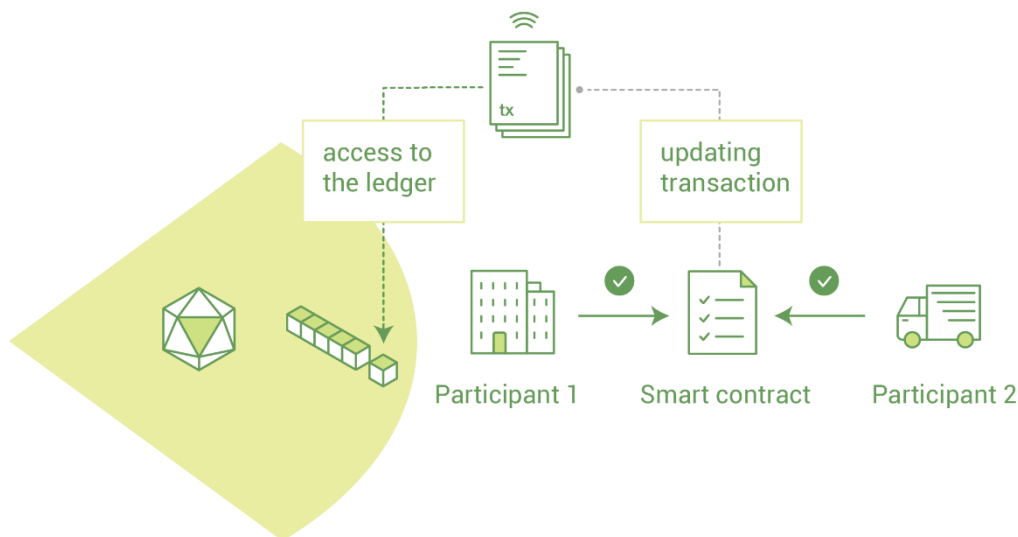
A blockchain ledger is often described as **decentralized** because it is replicated across many network participants, each of whom **collaborate** in its maintenance. We'll see that decentralization and collaboration are powerful attributes that mirror the way businesses exchange goods and services in the real world.



In addition to being decentralized and collaborative, the information recorded to a blockchain is append-only, using cryptographic techniques that guarantee that once a transaction has been added to the ledger it cannot be modified. This property of immutability makes it simple to determine the provenance of information because participants can be sure information has not been changed after the fact. It's why blockchains are sometimes described as **systems of proof**.

Smart Contracts

To support the consistent update of information – and to enable a whole host of ledger functions (transacting, querying, etc) – a blockchain network uses **smart contracts** to provide controlled access to the ledger.

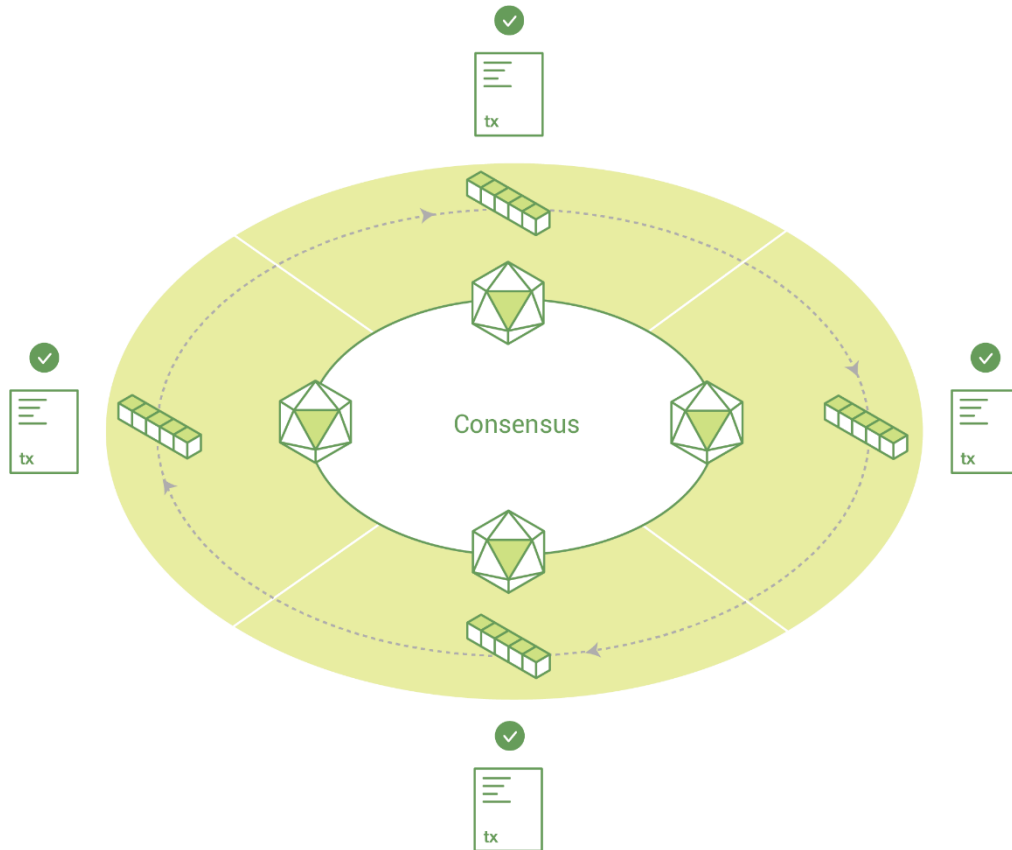


Smart contracts are not only a key mechanism for encapsulating information and keeping it simple across the network, they can also be written to allow participants to execute certain aspects of transactions automatically.

A smart contract can, for example, be written to stipulate the cost of shipping an item that changes depending on when it arrives. With the terms agreed to by both parties and written to the ledger, the appropriate funds change hands automatically when the item is received.

Consensus

The process of keeping the ledger transactions synchronized across the network – to ensure that ledgers only update when transactions are approved by the appropriate participants, and that when ledgers do update, they update with the same transactions in the same order – is called **consensus**.



We'll learn a lot more about ledgers, smart contracts and consensus later. For now, it's enough to think of a blockchain as a shared, replicated transaction system which is updated via smart contracts and kept consistently synchronized through a collaborative process called consensus.

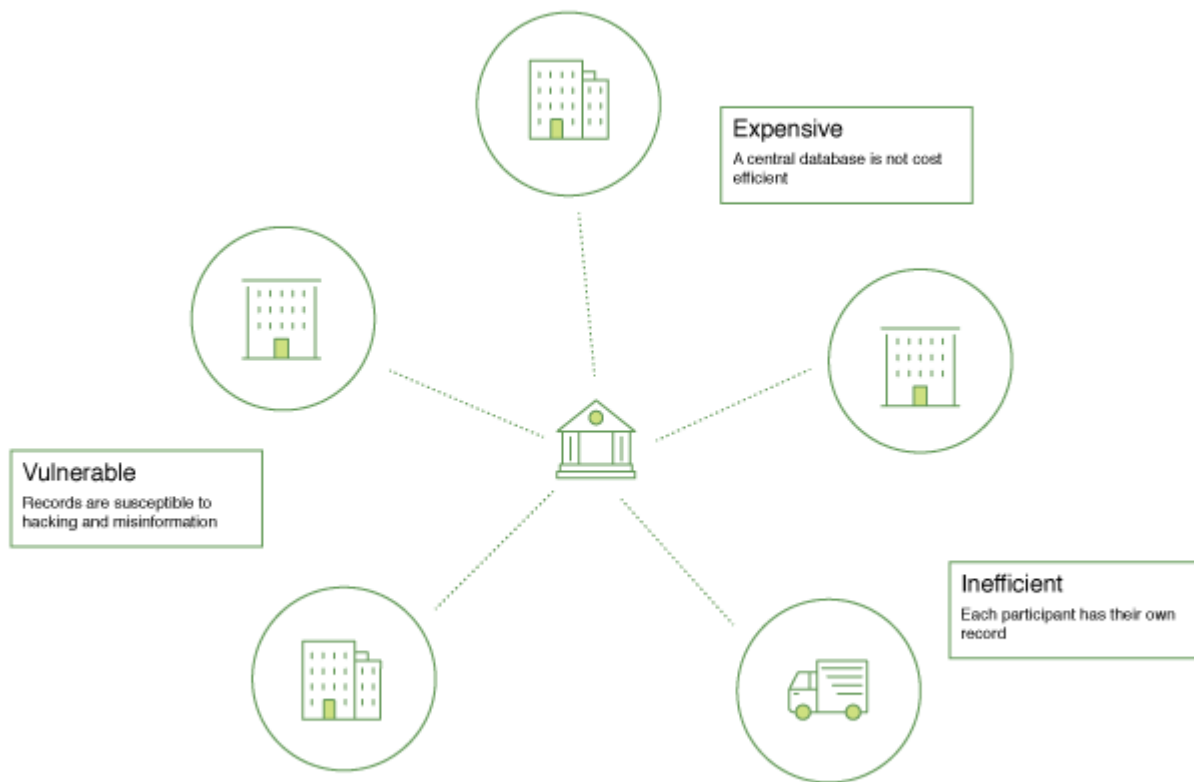
Why is a Blockchain useful?

Today's Systems of Record

The transactional networks of today are little more than slightly updated versions of networks that have existed since business records have been kept. The members of a **Business Network** transact with each other,

but they maintain separate records of their transactions. And the things they're transacting – whether it's Flemish tapestries in the 16th century or the securities of today – must have their provenance established each time they're sold to ensure that the business selling an item possesses a chain of title verifying their ownership of it.

What you're left with is a business network that looks like this:



Modern technology has taken this process from stone tablets and paper folders to hard drives and cloud platforms, but the underlying structure is the same. Unified systems for managing the identity of network participants do not exist, establishing provenance is so laborious it takes days to clear securities transactions (the world volume of which is numbered in the many trillions of dollars), contracts must be signed and executed manually, and every database in the system contains unique information and therefore represents a single point of failure.

It's impossible with today's fractured approach to information and process sharing to build a system of record that spans a business network, even though the needs of visibility and trust are clear.

The Blockchain Difference

What if instead of the rat's nest of inefficiencies represented by the “modern” system of transactions, business networks had standard methods for establishing identity on the network, executing transactions, and storing data? What if establishing the provenance of an asset could be determined by looking through a list of transactions that, once written, cannot be changed, and can therefore be trusted?

That business network would look more like this:



This is a blockchain network. Every participant in it has their own replicated copy of the ledger. In addition to ledger information being shared, the processes which update the ledger are also shared. Unlike today's systems, where a participant's **private** programs are used to update their **private** ledgers, a blockchain system has **shared** programs to update **shared** ledgers.

With the ability to coordinate their business network through a shared ledger, blockchain networks can reduce the time, cost, and risk associated with private information and processing while improving trust and visibility.

You now know what blockchain is and why it's useful. There are a lot of other details that are important, but they all relate to these fundamental ideas of the sharing of information and processes.

1.2 Hyperledger Fabric

The Linux Foundation founded Hyperledger in 2015 to advance cross-industry blockchain technologies. Rather than declaring a single blockchain standard, it encourages a collaborative approach to developing blockchain technologies via a community process, with intellectual property rights that encourage open development and the adoption of key standards over time.

Hyperledger Fabric is one of the blockchain projects within Hyperledger. Like other blockchain technologies, it has a ledger, uses smart contracts, and is a system by which participants manage their transactions.

Where Hyperledger Fabric breaks from some other blockchain systems is that it is **private** and **permissioned**. Rather than an open permissionless system that allows unknown identities to participate in the network (requiring protocols like Proof of Work to validate transactions and secure the network), the members of a Hyperledger Fabric network enroll through a **Membership Service Provider (MSP)**.

Hyperledger Fabric also offers several pluggable options. Ledger data can be stored in multiple formats, consensus mechanisms can be switched in and out, and different MSPs are supported.

Hyperledger Fabric also offers the ability to create **channels**, allowing a group of participants to create a separate ledger of transactions. This is an especially important option for networks where some participants might be competitors and not want every transaction they make - a special price they're offering to some participants and not others, for example - known to every participant. If two participants form a channel, then those participants - and no others - have copies of the ledger for that channel.

Shared Ledger

Hyperledger Fabric has a ledger subsystem comprising two components: the **world state** and the **transaction log**. Each participant has a copy of the ledger to every Hyperledger Fabric network they belong to.

The world state component describes the state of the ledger at a given point in time. It's the database of the ledger. The transaction log component records all transactions which have resulted in the current value of the world state. It's the update history for the world state. The ledger, then, is a combination of the world state database and the transaction log history.

The ledger has a replaceable data store for the world state. By default, this is a LevelDB key-value store database. The transaction log does not need to be pluggable. It simply records the before and after values of the ledger database being used by the blockchain network.

Smart Contracts

Hyperledger Fabric smart contracts are written in **chaincode** and are invoked by an application external to the blockchain when that application needs to interact with the ledger. In most cases chaincode only interacts with the database component of the ledger, the world state (querying it, for example), and not the transaction log.

Chaincode can be implemented in several programming languages. The currently supported chaincode language is [Go](#) with support for Java and other languages coming in future releases.

Privacy

Depending on the needs of a network, participants in a Business-to-Business (B2B) network might be extremely sensitive about how much information they share. For other networks, privacy will not be a top concern.

Hyperledger Fabric supports networks where privacy (using channels) is a key operational requirement as well as networks that are comparatively open.

Consensus

Transactions must be written to the ledger in the order in which they occur, even though they might be between different sets of participants within the network. For this to happen, the order of transactions must be established and a method for rejecting bad transactions that have been inserted into the ledger in error (or maliciously) must be put into place.

This is a thoroughly researched area of computer science, and there are many ways to achieve it, each with different trade-offs. For example, PBFT (Practical Byzantine Fault Tolerance) can provide a mechanism for file replicas to communicate with each other to keep each copy consistent, even in the event of corruption. Alternatively, in Bitcoin, ordering happens through a process called mining where competing computers race to solve a cryptographic puzzle which defines the order that all processes subsequently build upon.

Hyperledger Fabric has been designed to allow network starters to choose a consensus mechanism that best represents the relationships that exist between participants. As with privacy, there is a spectrum of needs;

from networks that are highly structured in their relationships to those that are more peer-to-peer. Hyperledger Fabric consensus mechanisms currently include SOLO, Kafka, and will soon extend to SBFT (Simplified Byzantine Fault Tolerance).

1.3 Hyperledger Composer

Hyperledger Composer is an extensive, open development toolset and framework to make developing blockchain applications easier. Our primary goal is to accelerate time to value, and make it easier to integrate your blockchain applications with the existing business systems. You can use Composer to rapidly develop use cases and deploy a blockchain solution in weeks rather than months. Composer allows you to model your business network and integrate existing systems and data with your blockchain applications.

Hyperledger Composer supports the existing [Hyperledger Fabric blockchain](#) infrastructure and runtime, which supports pluggable blockchain consensus protocols to ensure that transactions are validated according to policy by the designated business network participants.

Everyday applications can consume the data from business networks, providing end users with simple and controlled access points.

You can use Hyperledger Composer to quickly model your current business network, containing your existing assets and the transactions related to them; assets are tangible or intangible goods, services, or property. As part of your business network model, you define the transactions which can interact with assets. Business networks also include the participants who interact with them, each of which can be associated with a unique identity, across multiple business networks.

How does Hyperledger Composer work in practice?

For an example of a business network in action; a realtor can quickly model their business network as such:

- **Assets:** houses and listings
- **Participants:** buyers and homeowners
- **Transactions:** buying or selling houses, and creating and closing listings

Participants can have their access to transactions restricted based on their role as either a buyer, seller, or realtor. The realtor can then create an application to present buyers and sellers with a simple user interface for viewing open listings and making offers. This business network could also be integrated with existing inventory system, adding new houses as assets and removing sold properties. Relevant other parties can be registered as participants, for example a land registry might interact with a buyer to transfer ownership of the land.

CHAPTER 02

PROBLEM STATEMENT

To develop blockchain based support system for tracking liquid fuel distribution. Use a blockchain system for payment to speed up the process significantly & cut cost at the same time. Using blockchain platform- Hyperledger composer and fabric, various multiple parties involved in operations from upstream to downstream, such as gasoline producer, shipper, supplier, transporter, regulator & settlement provider are brought on to a single platform for enhancing trust & visibility. Also use smart contracts, to automate operations and provide data integrity throughout the supply chain management of gasoline. And make a user-friendly user interface for the gasoline supply chain system.

CHAPTER 03

LITERATURE REVIEW

- Supply chain relationships in the chemicals and petroleum industry are as vast and intricate as the geographies in which they operate. In the current “lower-for-longer” crude oil market, capitalizing on meaningful insight, cost take out, and the associated transformation opportunities for efficiency and longevity are keys to profitability. This applies across all internal and external supply chain functions for different subsidiary companies and divisions, and across established joint venture partners and supplier/subcontractor relationships. Each participant in a supply chain must maintain their records, updated with their transactions and systems, which must then be reconciled among other participants in the network. This time-consuming, manual reconciliation process is susceptible to errors or manipulation of transactions. As a result, all participants in the supply chain incur costs and delays associated with this reconciliation. These issues are further complicated when third-party validation or intermediaries have to be brought in to resolve disputes
- The potential uses for blockchain are growing, some of which could have significant implications for the oil industry. The proposed project overcomes its advantages over traditional systems, some of its potential applications, and how these can be applied to the oil and gas market, with a particular focus on taxation and compliance.

CHAPTER 04

MODEL DESIGN AND WORKING

4.1 Components of supply chain

Given geographic, political and operational realities, along with the scale of investment required within the industry, companies in chemicals and petroleum cannot exist in isolation. Most are connected beyond the industry consolidation we've witnessed recently through engineering, procurement, construction management, joint ventures, or other key service supplier relationships. These companies rely on extensive supply chain networks to bring their products and services to market. Their industry network connects suppliers, partners, distributors and customers, and operates across geographies and regulatory boundaries. Industry networks become more complex as the number of participants in them increases. They also reorganize constantly through restructuring initiatives, mergers, acquisitions or regulatory actions. Assets, goods and services are exchanged within these supply chain networks, and value is created through transactions governed by contracts. For a network to function properly, transactions must be recorded accurately in a system of record and be accessible to all participants. This system of record is often referred to as a ledger even though it is usually a digital record.

4.2 Key Concepts

Blockchain State Storage

All transactions submitted through a business network are stored on the blockchain ledger, and the current state of assets and participants are stored in the blockchain state database. The blockchain distributes the ledger and the state database across a set of peers and ensures that updates to the ledger and state database are consistent across all peers using a consensus algorithm.

Connection Profiles

Hyperledger Composer uses *Connection Profiles* to define the system to connect to. A connection profile is a JSON document that is part of a business network card. These profiles are usually provided by the creator of the system they refer to and should be used to create business network cards in order to be able to connect to that system.

Assets

Assets are tangible or intangible goods, services, or property, and are stored in registries. Assets can represent almost anything in a business network, for example, a house for sale, the sale listing, the land registry certificate for that house, and the insurance documents for that house may all be assets in one or more business networks.

Blockchain Support for Tracking Liquid

Assets must have a unique identifier, but other than that, they can contain whatever properties you define. Assets may be *related to* other assets or participants.

Participants

Participants are members of a business network. They may own assets and submit transactions. Participant types are modeled, and like assets, must have an identifier and can have any other properties as required. A participant can be mapped to one or multiple identities.

Identities

An identity is a digital certificate and private key. Identities are used to transact on a business network and must be mapped to a participant in the business network. A single identity is stored in a business network card and if that identity has been mapped to a participant, it allows the user of that business network card to transact on a business network as that participant.

Business Network cards

Business network cards are a combination of an identity, a connection profile, and metadata, the metadata optionally containing the name of the business network to connect to. Business network cards simplify the process of connecting to a business network, and extend the concept of an identity outside the business network to a 'wallet' of identities, each associated with a specific business network and connection profile.

Transactions

Transactions are the mechanism by which participants interact with assets. This could be as simple as a participant placing a bid on a asset in an auction, or an auctioneer marking an auction closed, automatically transferring ownership of the asset to the highest bidder.

Queries

Queries are used to return data about the blockchain world-state. Queries are defined within a business network, and can include variable parameters for simple customization. By using queries, data can be easily extracted from your blockchain network. Queries are sent by using the Hyperledger Composer API.

Events

Events are defined in the business network definition in the same way as assets or participants. Once events have been defined, they can be emitted by transaction processor functions to indicate to external systems that something of importance has happened to the ledger. Applications can subscribe to emitted events through the `composer-client` API.

Access Control

Business networks may contain a set of access control rules. Access control rules allow fine-grained control over what participants have access to what assets in the business network and under what conditions. The access control language is rich enough to capture sophisticated conditions declaratively, such as "only the owner of a

Blockchain Support for Tracking Liquid vehicle can transfer ownership of the vehicle". Externalizing access control from transaction processor function logic makes it easier to inspect, debug, develop and maintain.

Historian registry

The historian is a specialised registry which records successful transactions, including the participants and identities that submitted them. The historian stores transactions as `HistorianRecord` assets, which are defined in the Hyperledger Composer system namespace.

4.3 Architecture

Hyperledger Composer enables architects and developers to quickly create "full-stack" blockchain solutions. I.e. business logic that runs on the blockchain, REST APIs that expose the blockchain logic to web or mobile applications, as well as integrating the blockchain with existing enterprise systems of record.

Hyperledger Composer is composed of the following high-level components:

- Execution Runtimes
- JavaScript SDK
- Command Line Interface
- REST Server
- LoopBack Connector
- Playground Web User Interface
- Yeoman code generator
- VSCode and Atom editor plugins

Execution Runtimes

Hyperledger Composer has been designed to support different pluggable runtimes, and currently has three runtime implementations:

- Hyperledger Fabric v1.1. State is stored on the distributed ledger.
- Web, which executes within a web page, and is used by Playground. State is stored in browser local storage.
- Embedded, which executes within a Node.js process, and is used primarily for unit testing business logic. State is stored in an in-memory key-value store.

Connection Profiles

Connection Profiles are used across Hyperledger Composer to specify how to connect to an execution runtime. There are different configuration options for each *type* of execution runtime. For example, the connection profile

Blockchain Support for Tracking Liquid
for a Hyperledger Fabric v1.1 runtime will contain the TCP/IP addresses and ports for the Fabric peers, as well as cryptographic certificates etc.

Connection Profiles are part of Business Network cards.

JavaScript SDK

The Hyperledger Composer JavaScript SDK is a set of Node.js APIs that enables developers to create applications to manage and interact with deployed business networks.

The APIs are split between two npm modules:

1. `composer-client` used to submit transactions to a business network or to perform Create, Read, Update, Delete operations on assets and participants
2. `composer-admin` used to manage business networks (install, start, upgrade)

Details of all the APIs are available as JSDocs (see reference).

`composer-client`

This module would usually be installed as a local dependency of an application. It provides the API that is used by business applications to connect to a business network to access **assets**, **participants** and submitting **transactions**. When in production this is only module that needs to be added as a direct dependency of the application.

`composer-admin`

This module would usually be installed as a local dependency of **administrative** applications. This API permits the creation of and deployment of **business network definitions**.

Command Line Interface

The `composer` command line tool enables developers and administrators to deploy and managed business network definitions.

REST Server

The Hyperledger Composer REST Server automatically generates a Open API (Swagger) REST API for a business network. The REST Server (based on LoopBack technology) converts the Composer model for a business network into an Open API definition, and at runtime implements Create, Read, Update and Delete support for assets and participants and allows transactions to be submitted for processing or retrieved.

CHAPTER 05

WORKING

5.1 Creating a business network

Developers use Hyperledger Composer to digitize business networks. The business network is accessed by multiple participants in the network, some of which may be responsible for the maintenance (hosting) of the network itself, referred to as maintainers of the network.

Typically each maintainer of the network will run several peer nodes (for crash fault tolerance) and Hyperledger Fabric replicates the distributed ledger across the set of peer nodes.

Model

Developers work with business analysts to define the domain data model for the business network. The data model is expressed using the Composer Modeling Language and defines the structure of the resources that will be stored on the ledger, or processed as transactions.

Once the domain model is in place, developers can capture *smart contracts* as executable transaction processor functions, written in JavaScript.

Access Control

In parallel developers or technical analysts can define the access control rules for the business network, to enforce which participants have access to the data on the ledger and under which conditions.

Deploy

Developers package the models, scripts and access control rules into a deployable *Business Network Archive* and use command line tools to deploy the archive to a runtime for testing.

Test

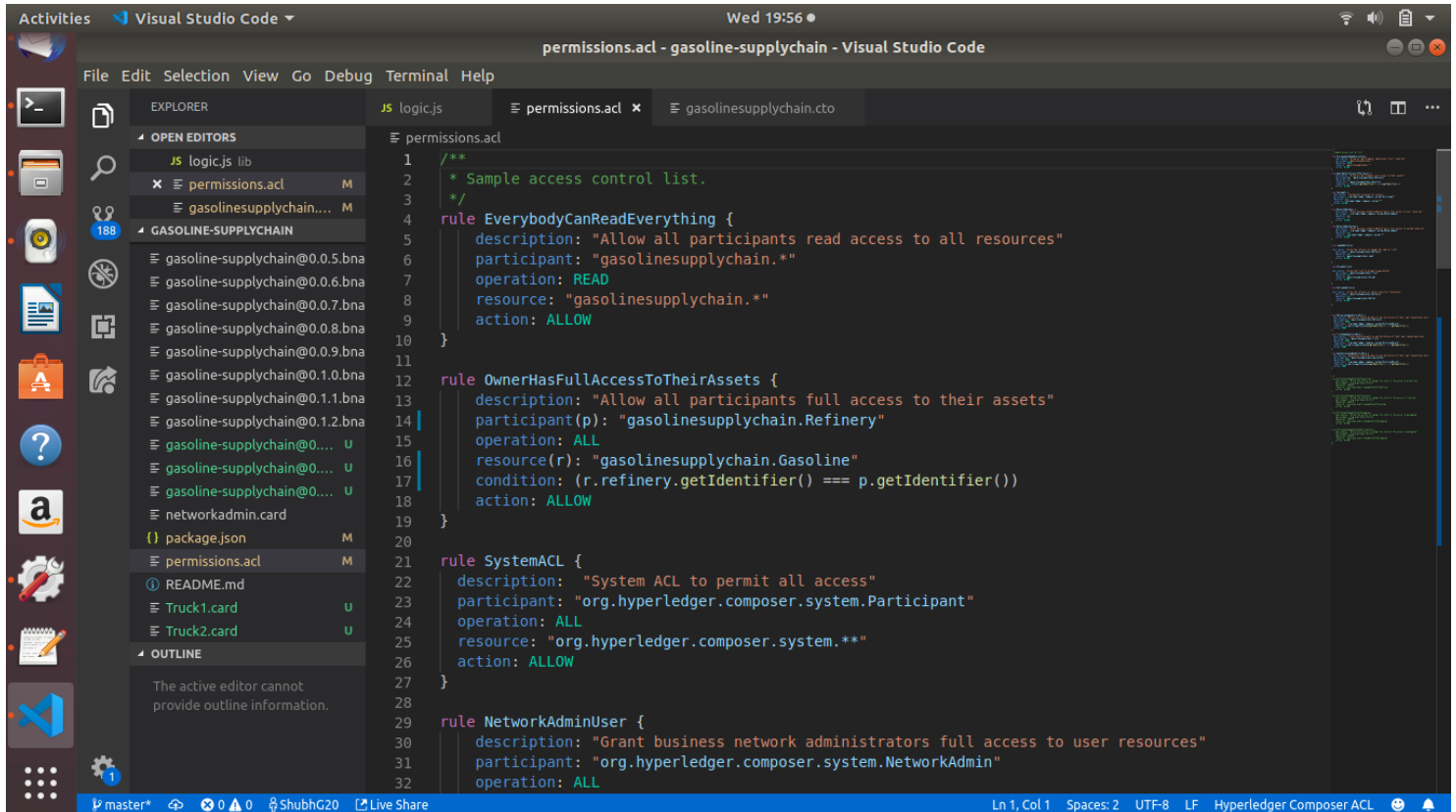
Like all business logic, it is important to create unit and system tests for business networks. Developers can use popular JavaScript testing frameworks such as Mocha and Chai to run unit tests (against the Node.js embedded runtime) or run system tests against a Hyperledger Fabric.

Integrate

Blockchain Support for Tracking Liquid

Once the business network is tested and in place, front-end applications need to be created. Use the REST Server to automatically generate a REST API for a business network, and then a skeleton generate Angular application using the Yeoman code generator.

The REST Server can be configured to authenticate the participants in the business network, ensuring that credentials and permissions are enforced.

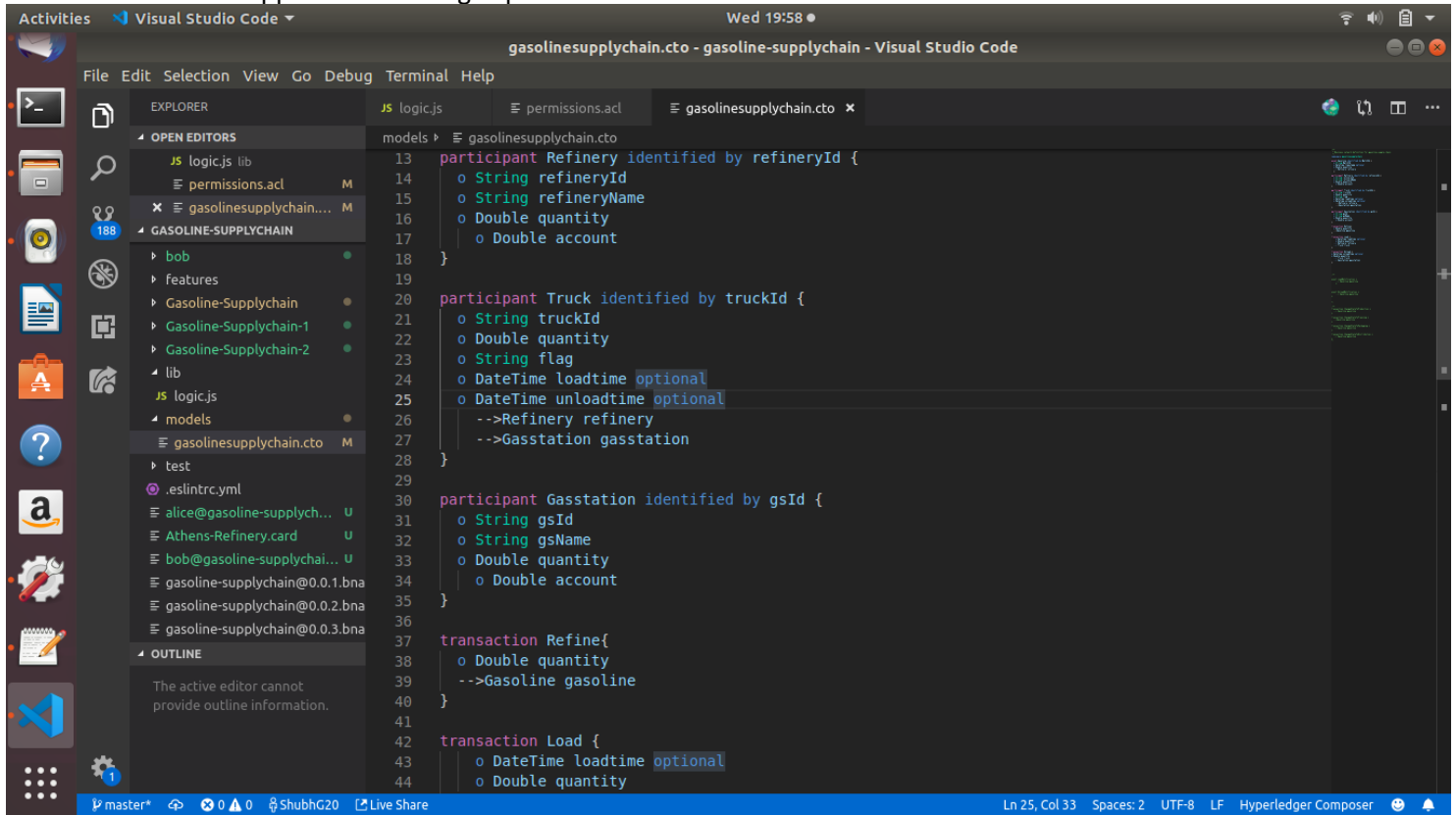


The screenshot shows the Visual Studio Code interface with the 'permissions.acl' file open in the editor. The file contains four ACL rules for the 'gasoline-supplychain' project. The Explorer sidebar on the left shows the project structure, including 'logic.js', 'permissions.acl', and 'gasolinesupplychain.cto'. The Outline sidebar is empty, showing a message: 'The active editor cannot provide outline information.' The status bar at the bottom indicates the file is at line 1, column 1, in UTF-8 encoding, with 2 spaces and LF line endings. The project is named 'master' and the user is 'ShubhG20'.

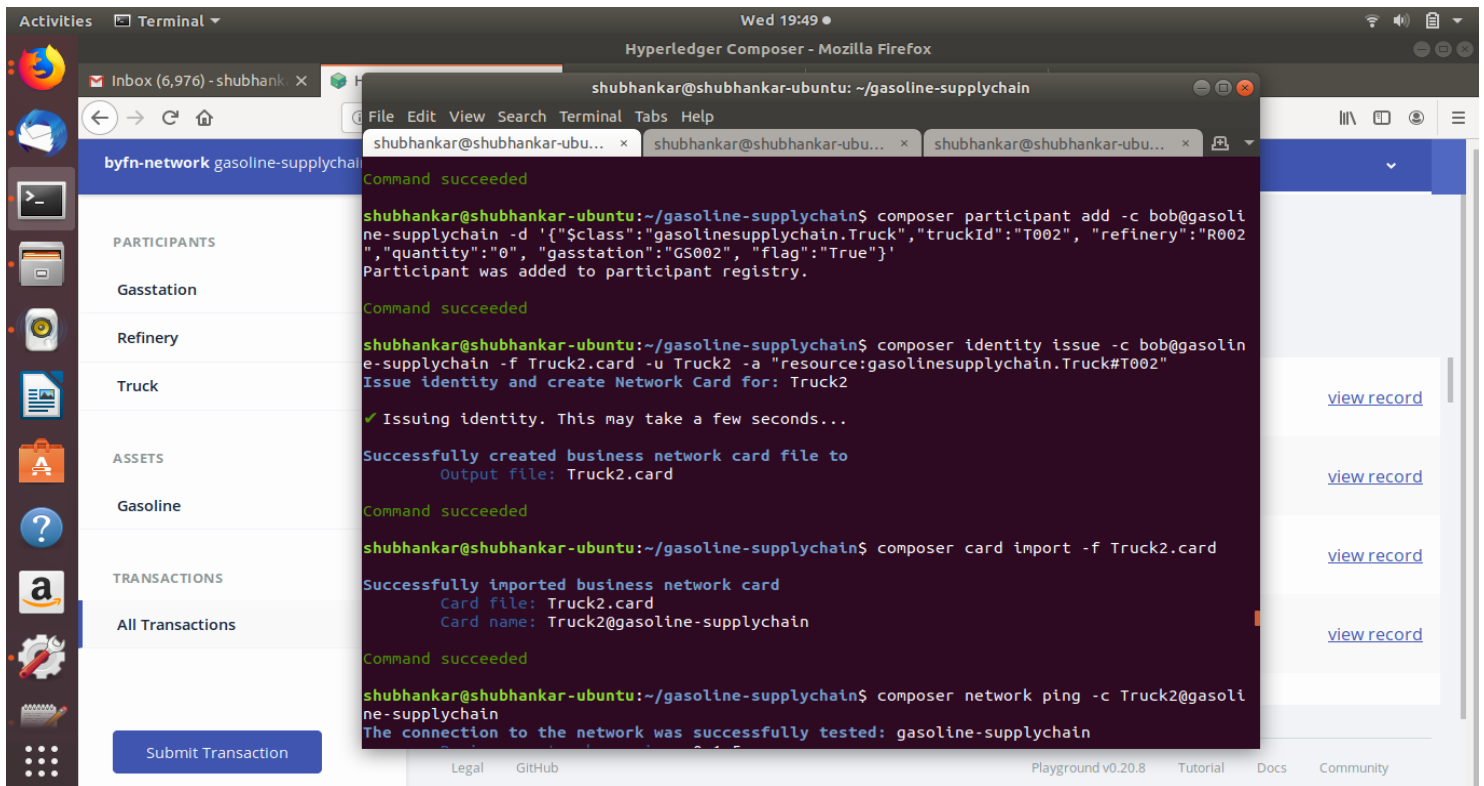
```
1 /**
2  * Sample access control list.
3  */
4  rule EverybodyCanReadEverything {
5      description: "Allow all participants read access to all resources"
6      participant: "gasolinesupplychain.*)"
7      operation: READ
8      resource: "gasolinesupplychain.*)"
9      action: ALLOW
10 }
11
12 rule OwnerHasFullAccessToTheirAssets {
13     description: "Allow all participants full access to their assets"
14     participant(p): "gasolinesupplychain.Refinery"
15     operation: ALL
16     resource(r): "gasolinesupplychain.Gasoline"
17     condition: (r.refinery.getIdentifier() === p.getIdentifier())
18     action: ALLOW
19 }
20
21 rule SystemACL {
22     description: "System ACL to permit all access"
23     participant: "org.hyperledger.composer.system.Participant"
24     operation: ALL
25     resource: "org.hyperledger.composer.system.*)"
26     action: ALLOW
27 }
28
29 rule NetworkAdminUser {
30     description: "Grant business network administrators full access to user resources"
31     participant: "org.hyperledger.composer.system.NetworkAdmin"
32     operation: ALL
```

Permission File

Blockchain Support for Tracking Liquid



Model File

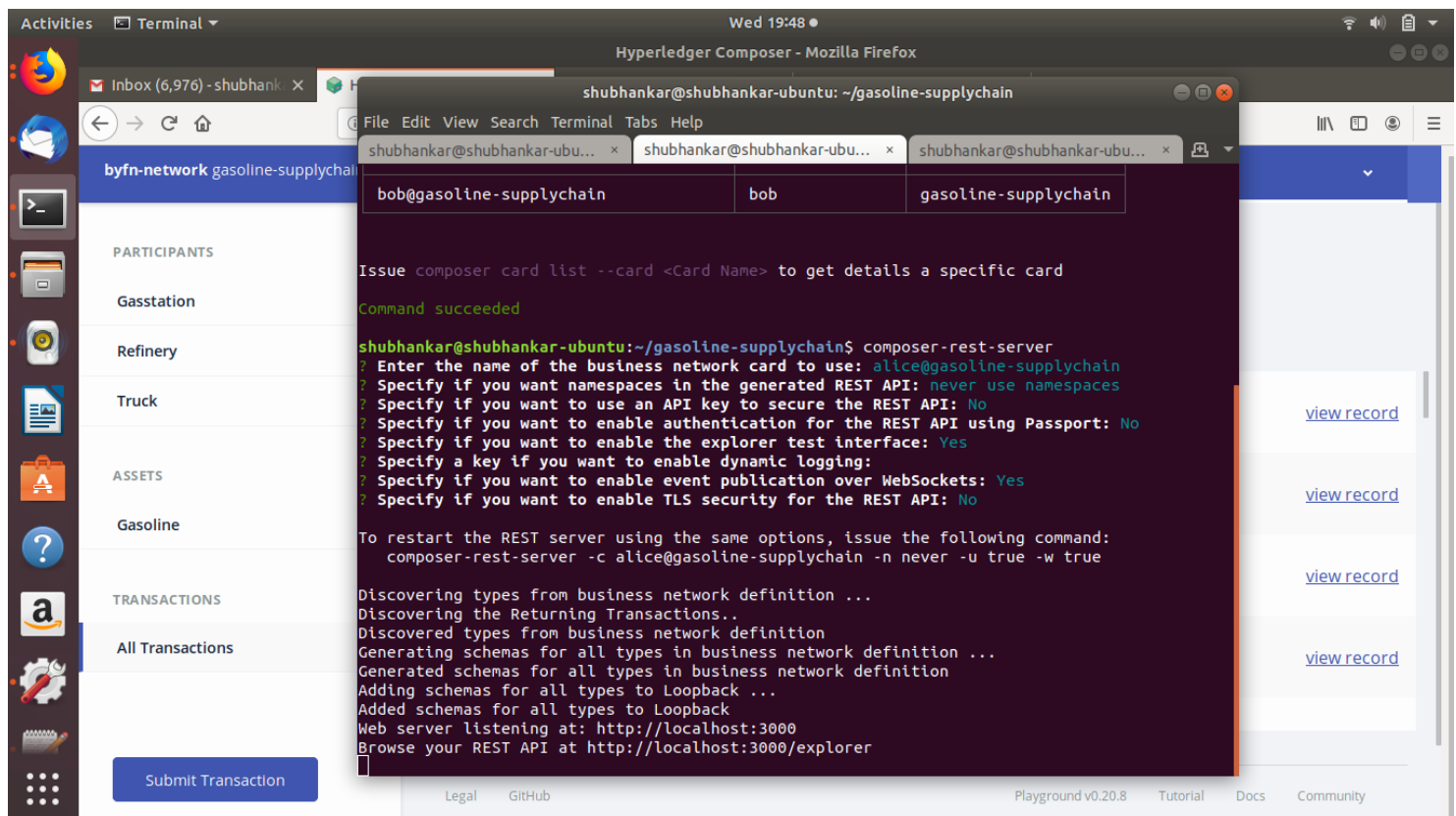


Creating Business Network Card

5.2 Deploying to rest api

Hyperledger Composer supports creating web, mobile or native Node.js applications. It includes the `composer-rest-server` (itself based on LoopBack technology) to automatically generate a REST API for a business network, and the `hyperledger-composer` code generation plugin for the Yeoman framework to generate a skeleton Angular application.

In addition it includes a rich set of JavaScript APIs to build native Node.js applications.



5.3 Angular app and composer playground

Blockchain Support for Tracking Liquid

shubhankar@shubhankar-ubuntu: ~/gasoline-supplychain

```
shubhankar@shubhankar-ubuntu:~/gasoline-supplychain$ composer card list
The following Business Network Cards are available:

Connection Profile: byfn-network
```

Card Name	UserId	Business Network
Athens-Refinery@gasoline-supplychain	Athens-Refinery	gasoline-supplychain
Truck1@gasoline-supplychain	Truck1	gasoline-supplychain
Truck2@gasoline-supplychain	Truck2	gasoline-supplychain
PeerAdmin@byfn-network-org2	PeerAdmin	
PeerAdmin@byfn-network-org1	PeerAdmin	
alice@gasoline-supplychain	alice	gasoline-supplychain
bob@gasoline-supplychain	bob	gasoline-supplychain

```
Issue composer card list --card <Card Name> to get details a specific card

Command succeeded

shubhankar@shubhankar-ubuntu:~/gasoline-supplychain$ composer-rest-server
? Enter the name of the business network card to use: alice@gasoline-supplychain
? Specify if you want namespaces in the generated REST API: never use namespaces
? Specify if you want to use an API key to secure the REST API: No
? Specify if you want to enable authentication for the REST API using Passport: No
```

List of Business Card

shubhankar@shubhankar-ubuntu: ~/gasoline-supplychain/Gasoline-Supplychain-2

```
shubhankar@shubhankar-ubuntu:~/gasoline-supplychain$ yo hyperledger-composer:angular
Welcome to the Hyperledger Composer Angular project generator
? Do you want to connect to a running Business Network? Yes
? Project name: Gasoline-Supplychain-2
? Description: Hyperledger Composer Angular project
? Author name: Shubhankar
? Author email: shubhankargaikwad2006@gmail.com
? License: Apache-2.0
? Name of the Business Network card: alice@gasoline-supplychain
? Do you want to generate a new REST API or connect to an existing REST API? Connect to an existing REST API
? REST server address: http://localhost
? REST server port: 3000
? Should namespaces be used in the generated REST API? Namespaces are not used
Created application!
Completed generation process
create app.js
create Dockerfile
create e2e/app.e2e-spec.ts
create e2e/app.po.ts
create e2e/tsconfig.e2e.json
create e2e/tsconfig.json
create karma.conf.js
create manifest.yml
create package.json
create protractor.conf.js
create proxy.conf.js
create README.md
create src/app/app-routing.module.ts
create src/app/app.component.css
create src/app/app.component.html
create src/app/app.component.spec.ts
```

Angular App

Blockchain Support for Tracking Liquid

The image displays two screenshots of the Hyperledger Composer Playground interface, showing the 'My Business Networks' section for a connection named 'byfn-network'. The interface is accessed via a Firefox browser at localhost:8080/login.

Top Screenshot: Shows three existing business network cards and a button to 'Deploy a new business network'.

Card Name	User ID	Business Network
PeerAdmin@byfn-network-o...	PeerAdmin	none
Athens-Refinery@gasoline-s...	Athens-Refinery	gasoline-supplychain
alice@gasoline-supplychain	alice	gasoline-supplychain

Bottom Screenshot: Shows four existing business network cards.

Card Name	User ID	Business Network
PeerAdmin@byfn-network-o...	PeerAdmin	none
Truck1@gasoline-supplychain	Truck1	gasoline-supplychain
Truck2@gasoline-supplychain	Truck2	gasoline-supplychain
bob@gasoline-supplychain	bob	gasoline-supplychain

Cards for Multiple Organization

Blockchain Support for Tracking Liquid

The screenshot displays the Hyperledger Composer web interface in a Mozilla Firefox browser. The interface is divided into a sidebar on the left and a main content area. The sidebar contains sections for PARTICIPANTS (Gasstation, Refinery, Truck), ASSETS (Gasoline), and TRANSACTIONS (All Transactions). The main content area shows a table of transactions with columns for Date, Time, Entry Type, and Participant. A 'Submit Transaction' button is located at the bottom left of the main area. A 'Historian Record' dialog box is open, showing a detailed view of a transaction.

Participants	Date, Time	Entry Type	Participant	Action
Truck	2019-07-24, 16:03:41	Unload	bob (NetworkAdmin)	view record
Gasoline	2019-07-24, 16:02:17	Load	bob (NetworkAdmin)	view record
Gasoline	2019-07-24, 16:01:22	Refine	bob (NetworkAdmin)	view record
All Transactions	2019-07-24, 15:59:51	Unload	alice (NetworkAdmin)	view record

Historian Record

Transaction Events (0)

```
1 {
2   "$class": "gasolinesupplychain.Unload",
3   "quantity": 25,
4   "truck": "resource:gasolinesupplychain.Truck#T002",
5   "gasstation": "resource:gasolinesupplychain.Gasstation#GS002",
6   "transactionId":
7     "5843644d8eef9e1f7e6c3c435748a1a249d4359438d19f25d7f6c0a7814fc2a3",
8   "timestamp": "2019-07-24T10:33:41.140Z"
9 }
```

Historian Records

Blockchain Support for Tracking Liquid

The screenshot displays the Hyperledger Composer REST server interface in a Mozilla Firefox browser. The browser has multiple tabs open, including 'Hyperledger Composer' and 'Gasoline-Supplychain-2'. The address bar shows 'localhost:3000/explorer/'. The main content area is titled 'Hyperledger Composer REST server' and lists several assets and transactions:

- Gasoline** : An asset named Gasoline
- Gasstation** : A participant named Gasstation
- Load** : A transaction named Load
- Refine** : A transaction named Refine
- Refinery** : A participant named Refinery
- System** : General business network methods
- Truck** : A participant named Truck
- Unload** : A transaction named Unload

Each item has links for 'Show/Hide', 'List Operations', and 'Expand Operations'. At the bottom, it indicates '[BASE URL: /api , API VERSION: 0.1.5]'.

The second screenshot shows the 'Gasoline-Supplychain-2' application interface. The browser tabs include 'Hyperledger Composer' and 'Gasoline-Supplychain-2'. The address bar shows 'localhost:4200/Gasoline'. The page title is 'Gasoline-Supplychain-2' with the subtitle 'gasoline-supplychain@0.1.5.bna'. There are tabs for 'Assets', 'Participants', and 'Transactions'. The 'Assets' tab is active, showing a list of 'Gasoline' assets. A '+ Create Asset' button is visible. Below the button is a table with the following data:

BatchId	timestamp	quantity	refinery	Actions
B001	2019-07-24T10:28:21.655Z	950	resource:gasolinesupplychain.Refinery#R001	
B002	2019-07-24T10:31:22.110Z	900	resource:gasolinesupplychain.Refinery#R002	

The bottom of the browser window shows the address bar with 'localhost:4200/Gasoline'.

Composer Rest Server

Blockchain Support for Tracking Liquid

The top screenshot shows the 'Gasoline-Supplychain-2' web application in a Firefox browser. The address bar shows 'localhost:4200/Refinery'. The application has three tabs: 'Assets', 'Participants', and 'Transactions'. The 'Participants' tab is active, and a dropdown menu is open showing 'Refinery' and 'Truck'. The 'Refinery' view displays a table with the following data:

refineryId	Gasstation	RefineryName	quantity	account	Actions
R001		Athens-Refinery	225	97500	
R002		Rafina-Refinery	350	92500	

The bottom screenshot shows the same application with the address bar at 'localhost:4200/Truck'. The 'Truck' view displays a table with the following data:

truckId	Gasstation	Refinery	unloadtime	gasstation	Actions
T001	300	True	2019-07-24T10:29:24.138Z	2019-07-24T10:29:51.022Z	resource:gasolinesupplychain.Refi... resource:gasolinesupplychain.Gas...
T002	25	True	2019-07-24T10:32:17.844Z	2019-07-24T10:33:41.140Z	resource:gasolinesupplychain.Refi... resource:gasolinesupplychain.Gas...

Adding Participants

Blockchain Support for Tracking Liquid

The top screenshot shows the 'Gasoline-Supplychain-2' application in a Firefox browser window. The address bar shows 'localhost:4200/Gasstation'. The application has three tabs: 'Assets', 'Participants', and 'Transactions'. The 'Participants' tab is active, showing a table with columns 'gsld', 'quantity', 'account', and 'Actions'. The table contains two rows: 'GS001' with quantity 25 and account 47500, and 'GS002' with quantity 25 and account 37500. A '+ | Create Participant' button is visible on the right.

gsld	quantity	account	Actions
GS001	25	47500	
GS002	25	37500	

The bottom screenshot shows the same application with the 'Transactions' tab active. A 'Refine' dialog box is open, showing a list of attributes: 'quantity' and 'gasoline'. The 'Refine' button is highlighted. The 'Invoke' button is visible at the bottom right of the dialog.

Performing Transactions

CHAPTER 06

CONCLUSIONS

Conclusions

Blockchain has the potential to generate substantial benefits for companies in the chemicals and petroleum industry. We understand the complexities of supply chain relationships in the chemicals and petroleum industry. Blockchain solutions create an opportunity to become more productive now and derive even greater benefits with the imminent increase in adoption. Blockchain technologies enable faster, permissioned, immutable, transparent and auditable business-to-business transactions among participants in the network and their suppliers, distributors and partners. Chemicals and petroleum industry executives need to understand how best to extract value from blockchain technologies and develop an adoption strategy. Blockchain adoption across the chemicals and petroleum industry and strategic ecosystem suppliers has the potential to transform how companies work across operations on a global industry scale, delivering meaningful value for every participant in the supply chain network.

CHAPTER 07

REFERENCES

References

1. <https://www.ibm.com/industries/oil-gas/blockchain>
2. <https://hyperledger.github.io/composer/v0.19/index.html>
3. <https://hyperledger-fabric.readthedocs.io/en/release-1.4/>
4. <https://medium.com/coinmonks/building-a-blockchain-application-using-hyperledger-fabric-with-angular-frontend-part-2-22ef7c77f53>
5. <https://medium.com/@wahabjawed/hyperledger-fabric-on-multiple-hosts-a33b08ef24f>

