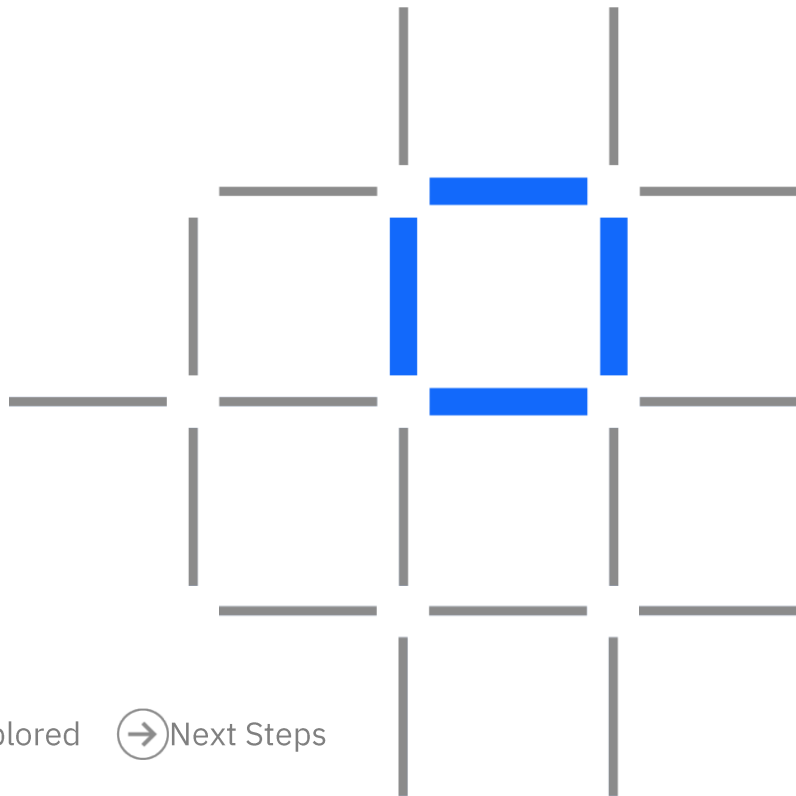


Blockchain Composed

A Technical Introduction to Hyperledger Composer

IBM Blockchain



Blockchain education series

Explained

Solutions

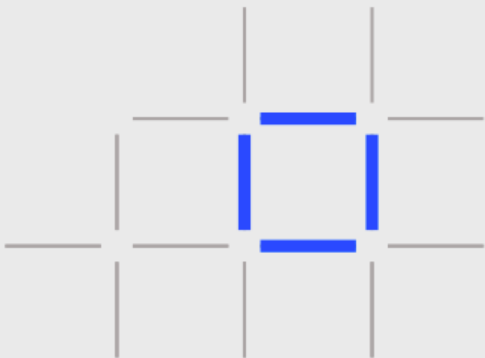
 **Composed**

 Architected

 Explored

 Next Steps

Contents



What is Hyperledger
Composer



Application Development

Writing the application

Modeling the business network



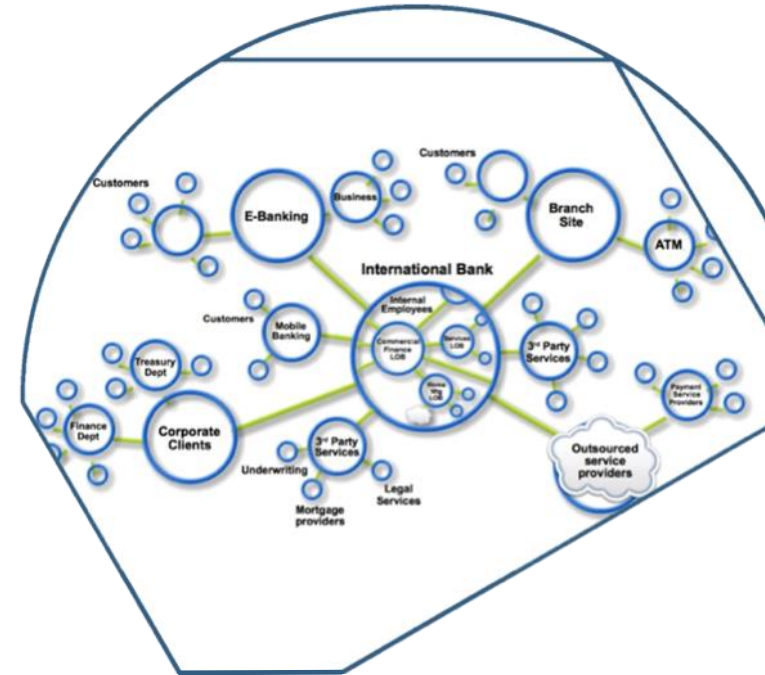
Effective Administration

Deploying to a blockchain

Interacting with systems of record

Blockchain Recap

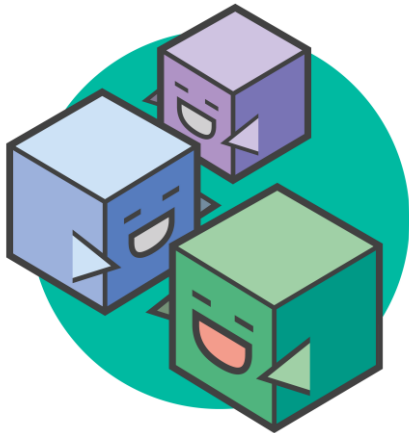
- Blockchain builds on basic business concepts
 - **Business Networks** connect businesses
 - **Participants** with Identity
 - **Assets** flow over business networks
 - **Transactions** describe asset exchange
 - **Contracts** underpin transactions
 - The **ledger** is a log of transactions
- Blockchain is a shared, replicated ledger
 - Consensus, immutability, finality, provenance



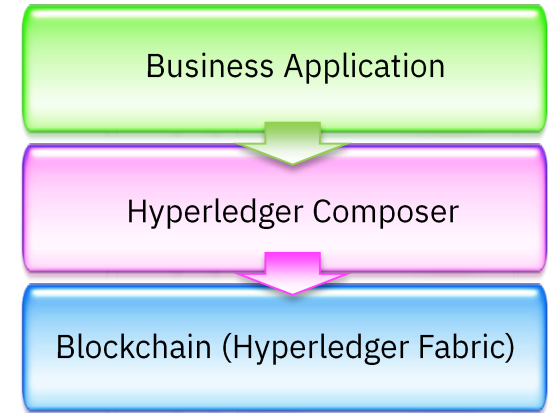
Hyperledger Composer: Accelerating time to value

<https://hyperledger.github.io/composer/>

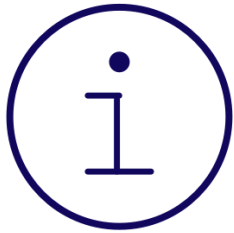
- A suite of high level application **abstractions** for business networks
- Emphasis on business-centric vocabulary for quick solution creation
- Reduce risk, and increase understanding and flexibility



- Features
 - Model your business networks, test and expose via APIs
 - Applications invoke APIs transactions to interact with business network
 - Integrate existing systems of record using loopback/REST
- Fully open and part of Linux Foundation Hyperledger
- Try it in your web browser now: <http://composer-playground.mybluemix.net/>



Benefits of Hyperledger Composer



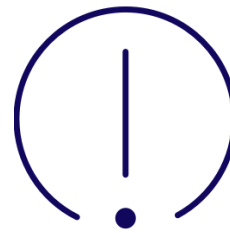
Increases understanding

Bridges simply from
business concepts to
blockchain



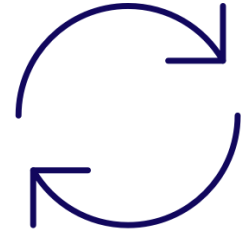
Saves time

Develop blockchain
applications more
quickly and cheaply



Reduces risk

Well tested, efficient
design conforms to
best practice



Increases flexibility

Higher level
abstraction makes it
easier to iterate

Extensive, Familiar, Open Development Toolset

```
asset Animal identi
  o String animal
  o AnimalType sp
  o MovementStatu
  o ProductionTyp
```

Data modelling



JavaScript
business logic



Web playground

```
composer-client
composer-admin
```



Client libraries



Editor support

```
$ composer
```

CLI utilities



Code generation

Powered by



LoopBack
Node.js Framework



Swagger

Existing systems and
data

User Roles in a Blockchain Project



Application Developer

- Developing the application that interacts with the ledger
- Modelling the business network
- Implementing the script files that define transaction behaviour



Solution Administrator

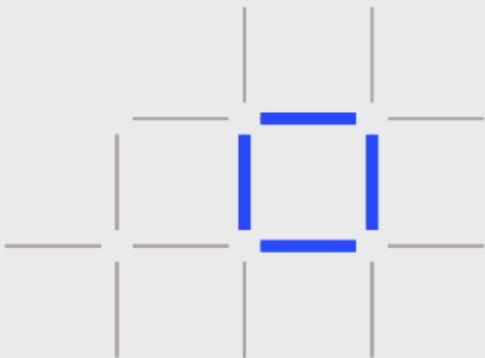
- Provisioning the target environment
- Deploying the business application
- Managing the blockchain



Business Network Participant

- Running an end-user application that invokes transactions
- Aware of business concepts: assets, participants and transactions
- May not be aware of blockchain underpinnings

Contents



What is Hyperledger
Composer



Application Development

Writing the application

Modeling the business network

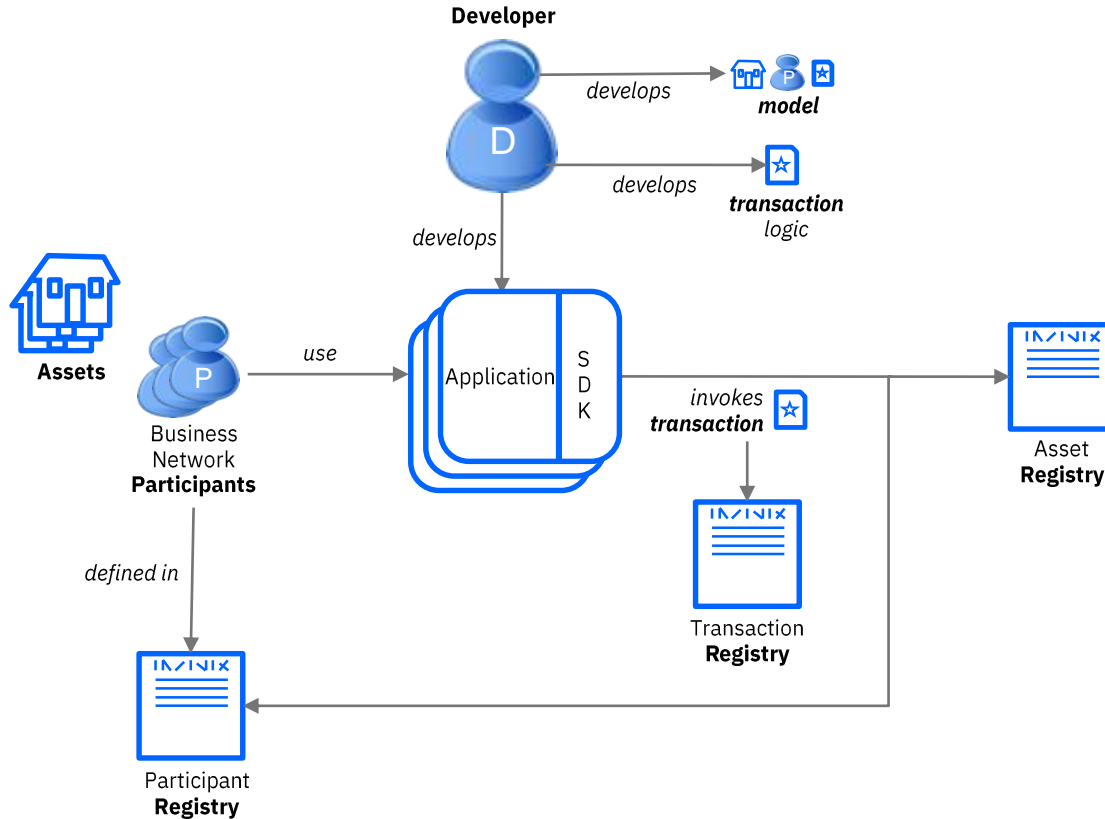


Effective Administration

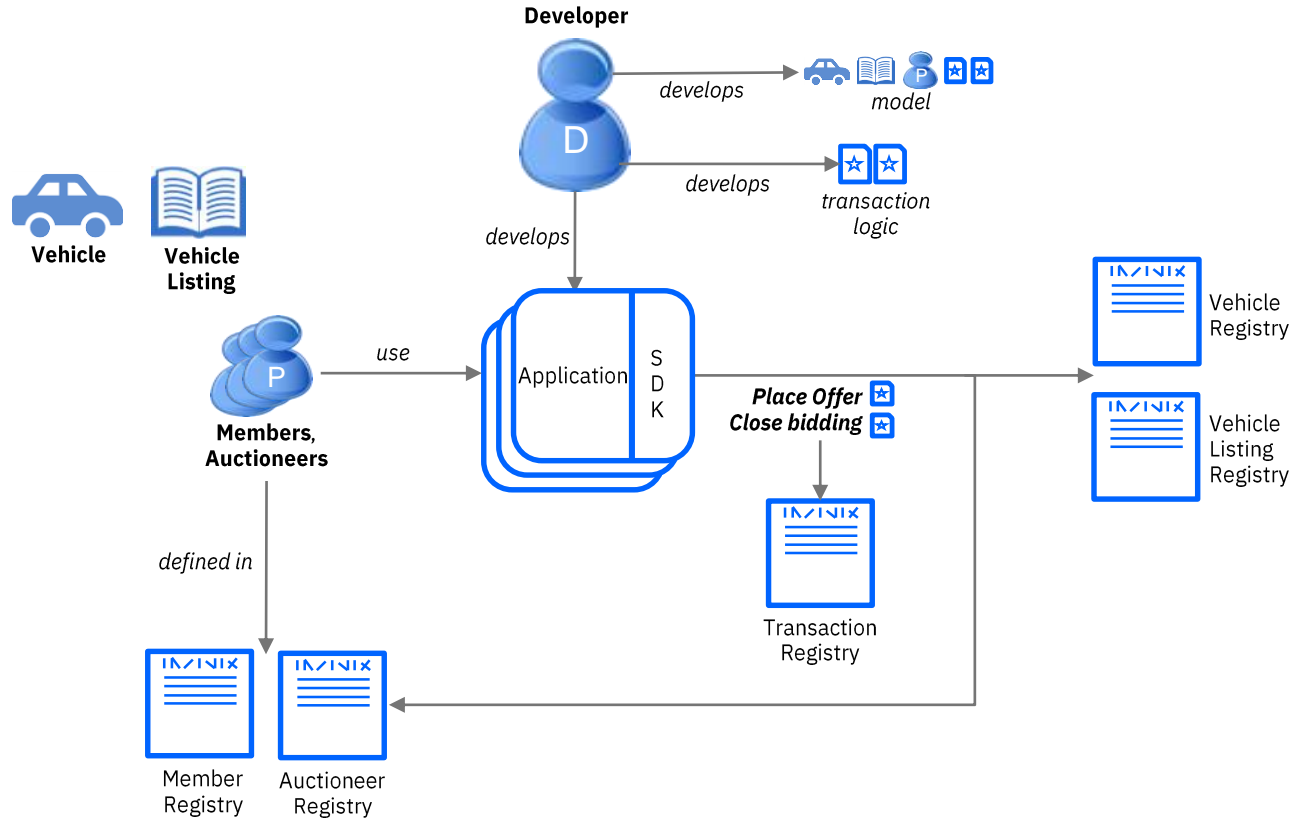
Deploying to a blockchain

Interacting with systems of record

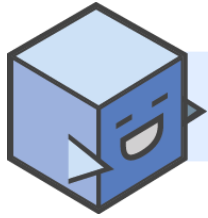
Key Concepts for the Application Developer



Example: Vehicle Auction

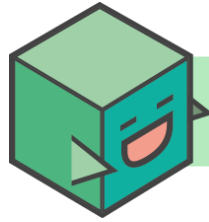


Developer Concepts



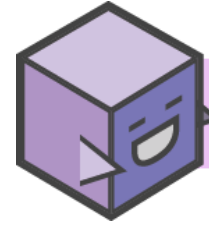
Applications

- Provides front-end for the user
 - May require different applications per participant
- Interacts with the registries
 - Add, delete, update, query
 - Registries persisted on blockchain
- Connects to blockchain via JavaScript client libraries (SDK) or REST



Models

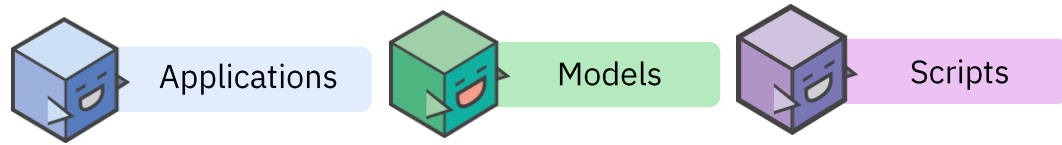
- A domain specific language (.CTO) that defines the type structure of
 - Assets
 - Participants
 - Transactions
- Aims to match how we talk about business networks in the real world



Scripts

- Scripts provide the implementation of transaction processor logic
- Specified in Javascript
- Designed for any reasonable Javascript developer to pick up easily

Vehicle Auction



Vehicle

VIN

(References a) Member

Vehicle Listing

listingId, reservePrice,
description, state, offers[]
(References a) Vehicle



Vehicle



Vehicle Listing

User

email, firstName, lastName

Member (extends User)
balance

Auctioneer (extends User)



Members,
Auctioneers

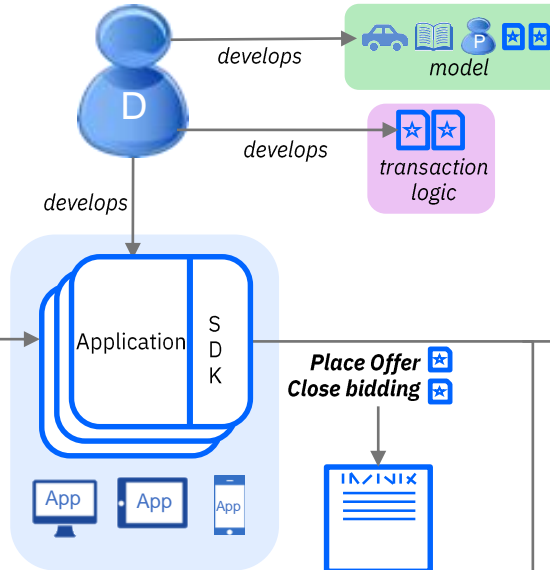
defined in



Member Registry



Auctioneer Registry



Place Offer

bidPrice

(References a) listing, member

Close Bidding

(References a) listing



Vehicle Registry



Vehicle Listing Registry

Place Offer

If listing is for sale, add offer to this vehicle listing's offers[]

Close Bidding

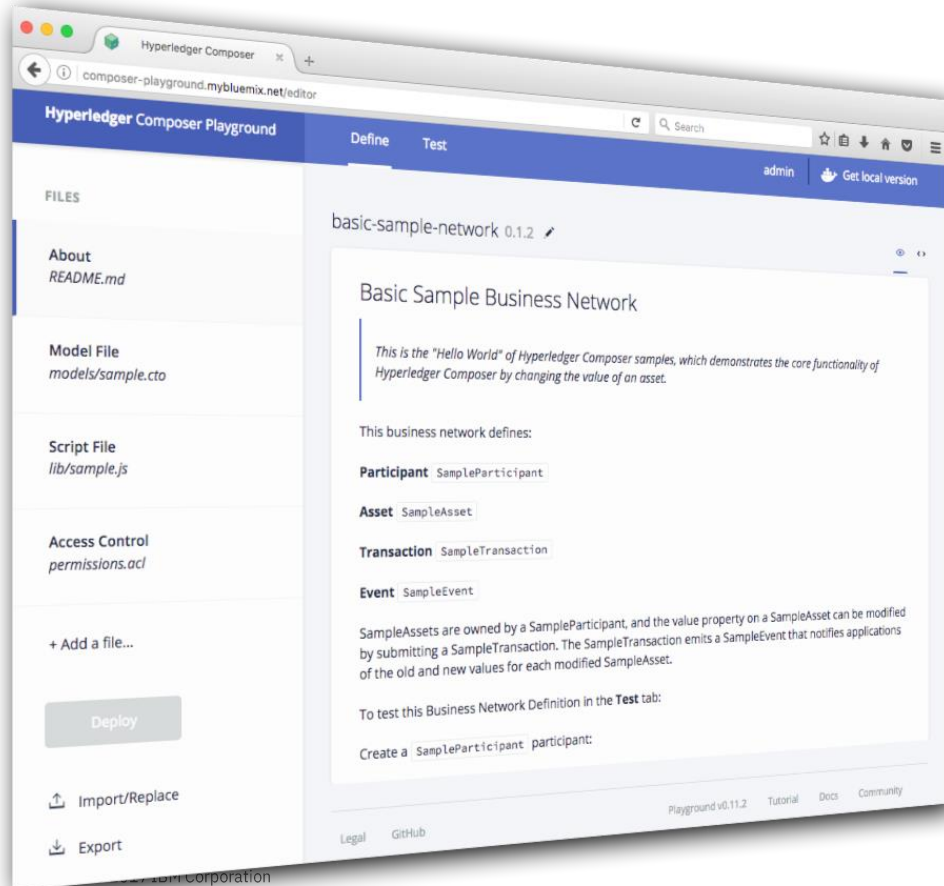
If reserve price met

Increment seller's balance

Decrement buyer's balance

Change owner to buyer

Tools: Composer Playground



- Web tool for defining and testing Hyperledger Composer models and scripts
- Designed for the application developer
 - Define assets, participants and transactions
 - Implement transaction processor scripts
 - Test by populating registries and invoking transactions
- Deploy to instances of Hyperledger Fabric V1, or simulate completely within browser
- Install on your machine or run online at <http://composer-playground.mybluemix.net>

Visual Studio Code Support

- Composer extension available for this popular tool
- Features to aid rapid Composer development
 - Edit all Composer file types with full syntax highlighting
 - Validation support for models, queries and ACLs
 - Inline error reporting
 - Snippets (press Ctrl+Space for code suggestions)
 - Generate UML diagrams from models
- Install directly from Code Marketplace

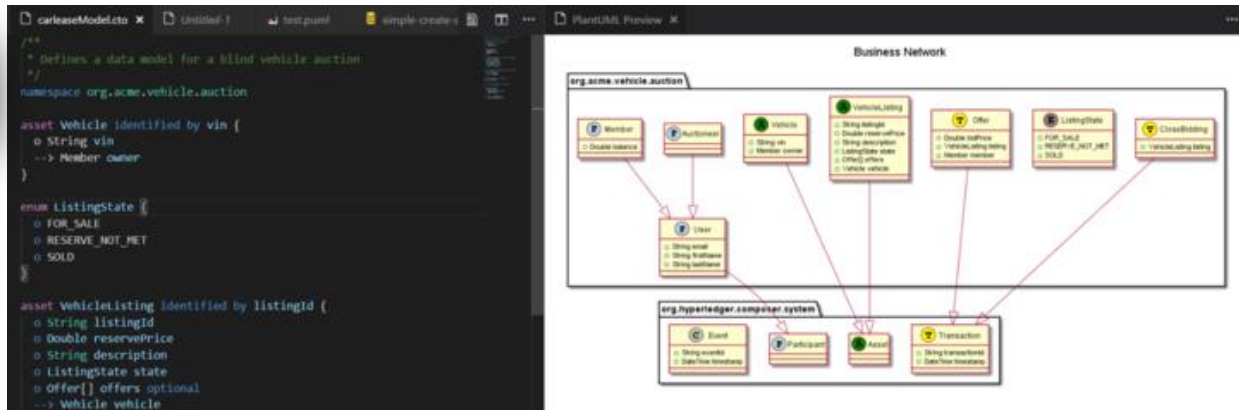
```
namespace org.acme.vehicle.lifecycle

import composer.base.Person
import composer.business.Business

participant PrivateOwner identified by email extends Person {
  o String email
}
```

```
[Composer] IllegalModelException: Could not find super type Person
participant PrivateOwner identified by email extends Person {
  o String email
}
```

Hyperledger Composer 0.11.3
Hyperledger Composer syntax highlighting
Hyperledger Composer VS Code Plugin



Creating the Application

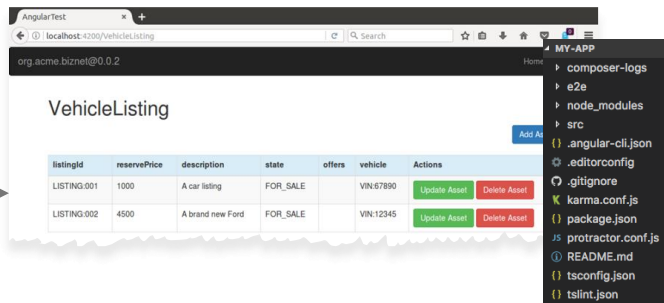
- JavaScript applications *require()* the NPM “composer-client” module
 - This provides the API to access assets, participants and transactions
- RESTful API (via Loopback) also available... see later
- Command-line tool available to generate skeleton command-line or Angular2 application from model
- Can also generate unit tests to help ensure quality code

```
const BusinessNetworkConnection = require('composer-client')
    .BusinessNetworkConnection;

this.bizNetworkConnection = new BusinessNetworkConnection();
this.titlesRegistry = this.bizNetworkConnection.getAssetRegistry
    ('net.biz.digitalPropertyNetwork.LandTitle')

let landTitle1 = factory.newResource
    ('net.biz.digitalPropertyNetwork', 'LandTitle', 'LID:1148');
landTitle1.owner = ownerRelation;
landTitle1.information = 'A nice house in the country';
this.titlesRegistry.add(landTitle1);
```

```
yo hyperledger-composer
```



Events and Queries

- Events allow applications to take action when a transaction occurs
 - Events are **defined** in **models**
 - Events are **emitted** by **scripts**
 - Events are **caught** by **applications**
- Caught events include transaction ID and other relevant information
- Queries allow applications to perform complex registry searches
 - They can be statically defined in a separate .qry file or generated dynamically by the application
 - They are invoked in the application using *buildQuery()* or *query()*
 - Queries require the blockchain to be backed by CouchDB

```
event SampleEvent {  
  --> SampleAsset asset  
  o String oldValue  
  o String newValue  
}
```

```
// Emit an event for the modified asset.  
var event = getFactory().newEvent('org.acme.sample', 'SampleEvent');  
event.asset = tx.asset;  
event.oldValue = oldValue;  
event.newValue = tx.newValue;  
emit(event);
```

```
businessNetworkConnection.on('SampleEvent', (event) => {  
  console.log(event);  
})
```

```
query selectCommoditiesWithHighQuantity {  
  description: "Select commodities based on quantity"  
  statement:  
    | SELECT org.acme.trading.Commodity  
    | WHERE (quantity > 60)  
}
```

```
return query('selectCommoditiesWithHighQuantity', {})
```


Access Control

```
rule EverybodyCanReadEverything {  
  description: "Allow all participants read access to all resources"  
  participant: "org.acme.sample.SampleParticipant"  
  operation: READ  
  resource: "org.acme.sample.*"  
  action: ALLOW  
}
```

```
rule OwnerHasFullAccessToTheirAssets {  
  description: "Allow all participants full access to their assets"  
  participant(p): "org.acme.sample.SampleParticipant"  
  operation: ALL  
  resource(r): "org.acme.sample.SampleAsset"  
  condition: (r.owner.getIdentifier() == p.getIdentifier())  
  action: ALLOW  
}
```

```
rule SystemACL {  
  description: "System ACL to permit all access"  
  participant: "org.hyperledger.composer.system.Participant"  
  operation: ALL  
  resource: "org.hyperledger.composer.system.*"  
  action: ALLOW  
}
```

- It is possible to restrict which resources can be read and modified by which participants
 - Rules are defined in an .acl file and deployed with the rest of the model
 - Transaction processors can also look up the current user and implement rules programmatically
- ACL rules can be simple (e.g. everybody can read all resources) or more complex (e.g. only the owner of an asset can do everything to it)
- Application supplies credentials (userid/secret) of the participant when connecting to the Fabric network
 - This also applies to Playground!
 - Remember to grant System ACL all access if necessary

Debugging

- Playground Historian allows you to view all transactions
 - See what occurred and when
- Diagnostics framework allows for application level trace
 - Uses the *Winston* Node.js logging framework
 - Application logging using DEBUG env var
 - Composer Logs sent to stdout and `./logs/trace_<processid>.trc`
- Fabric chaincode tracing also possible (see later)
- More information online:

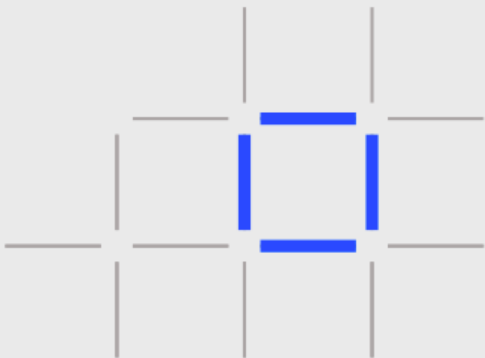
<https://hyperledger.github.io/composer/problems/diagnostics.html>

The screenshot displays the 'Default Historian Registry' interface. It features a table with columns: ID, Time, Participant ID, and Transaction Type. Three transactions are listed, each with a 'view data' link. An overlay window titled 'Transaction Data' is open, showing a JSON representation of a transaction record.

ID	Time	Participant ID	Transaction Type
af9faafd-d973-4647-9fad-0f58c0b...	17:15:00	emma	Offer
74e63603-7c7f-4bf2-b917-4c9707...	17:14:34	<system>	ActivateCurrentIdentity
e5a03410-7ead-46bc-a71f-588be...	17:14:30	matt	AddAsset

```
1 {
2   "$class": "org.hyperledger.composer.system.HistorianRecord",
3   "transactionId": "af9faafd-d973-4647-9fad-0f58c0ba7d15",
4   "transactionType": "Offer",
5   "transactionInvoked":
6     "resource:org.hyperledger.composer.system.Transaction#af9faafd-
7       d973-4647-9fad-0f58c0ba7d15",
8     "participantInvoking":
9       "resource:org.hyperledger.composer.system.Participant#emma",
10    "identityUsed":
11      "resource:org.hyperledger.composer.system.Identity#8d0fdf5ef7c0062f
12        67853ecf9b36544b2e2c36f0e9b9536166dc0f056a62a032",
13    "eventsEmitted": [],
14    "transactionTimestamp": "2017-08-11T16:15:00.161Z"
15 }
```

Contents



What is Hyperledger
Composer



Application Development

Writing the application

Modeling the business network

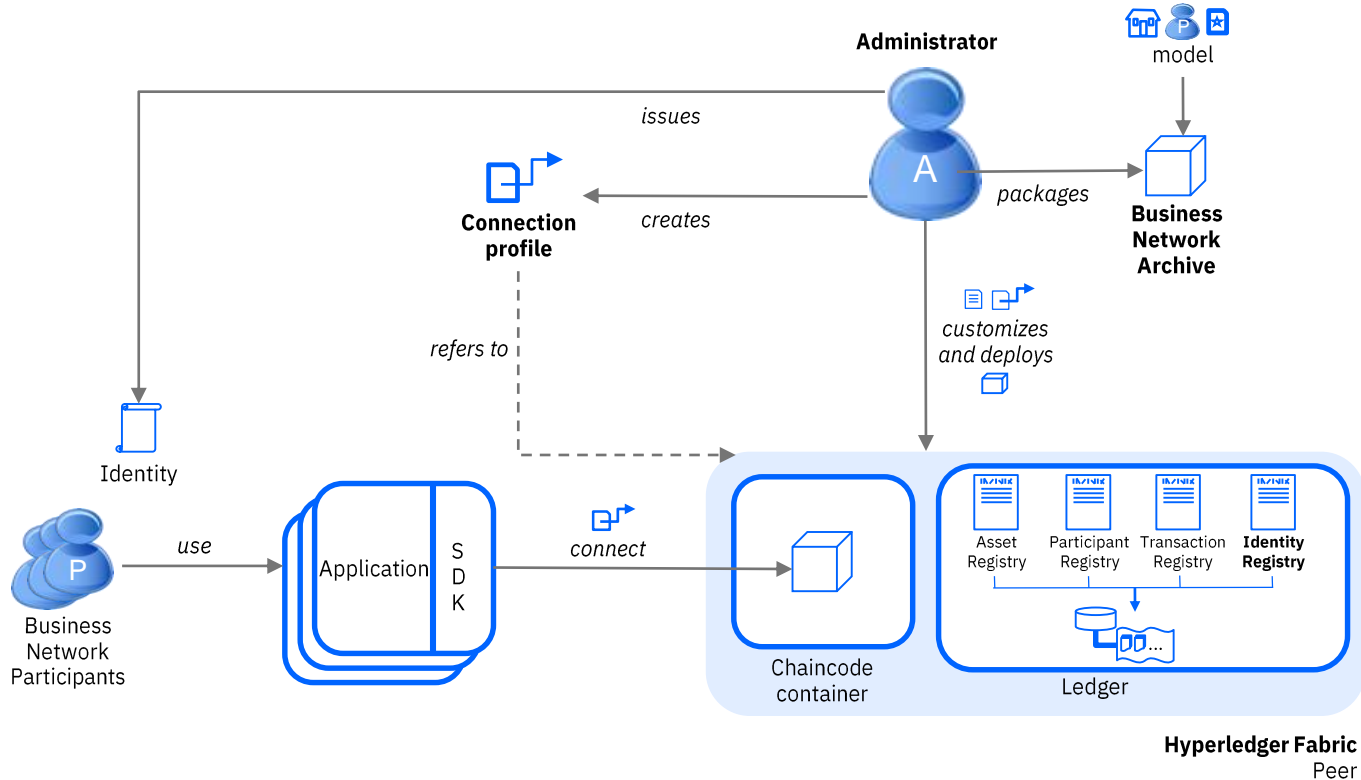


Effective Administration

Deploying to a blockchain

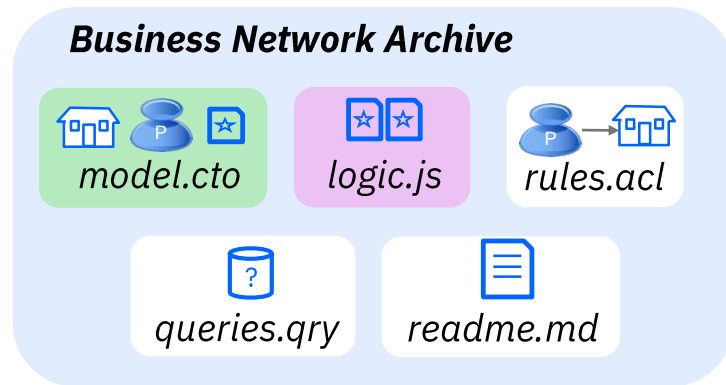
Interacting with systems of record

Key Concepts for the Administrator



Resources are packaged into BNA files

- Business Network Archive (.BNA) is a package of the resources used by Fabric:
 - Model files (.CTO)
 - Transaction processors (.JS)
 - Access Control Lists (.ACL)
 - Static queries (.QRY)
 - Documentation and versioning (.MD)
 - It does *not* contain the client application
- The BNA simplifies deployment of blockchain and promotion between environments
 - c.f. TAR, WAR, EAR, JAR, BAR...
- Create BNA files from Playground or command line
 - Build from filesystem or NPM module



```
composer archive create --archiveFile my.bna  
--sourceType module --sourceName myNetwork
```

Deployment to Hyperledger Fabric

The screenshot shows the 'Basic Configuration' tab of the Hyperledger Composer interface. It contains the following fields:

- Connection Profile Name:** hifabric
- Orderer(s):**
 - Orderer URL:** grpc://orderer.example.com:7050
- Channel:** composerchannel
- MSP ID:** Org1MSP
- Certificate Authority (CA):**
 - URL:** http://ca.org1.example.com:7054
 - Name:** ca.org1.example.com
- Peer(s):**
 - Peer Request URL:** grpc://peer0.org1.example.com:7051
 - Peer Event URL:** grpc://peer0.org1.example.com:7053
- Key Value Store Directory:** /home/composer/.composer-credentials

At the bottom, there are two buttons: 'Use this profile' (blue) and 'Export Connection Profile' (white with a download icon).

- Command line tool to script deployment

```
composer network deploy -p myProfile -a my.bna -i user -s secret
```

- Use Connection profiles to describe Fabric connection parameters
 - Export from Playground, generate from script or create by hand
- Enrollment in Hyperledger Fabric network required
 - Issue Fabric identity from Composer participants
- Additional command line options for management of business network
 - For example: download, list, start, undeploy, upgrade...

Participant Identity

- Participants require an *identity* in order to connect to Hyperledger Fabric
 - Issued by the administrator as a Hyperledger Fabric userid/secret
 - Supplied by the participant when the client application connects
- Composer Participant to Fabric Identity mapping is stored on the blockchain in an *identity registry*
- Perform identity management from Playground, Javascript, REST or command line
 - For example: Test connection, issue identity, bind an identity to a participant, revoke an identity, list identities

The 'Issue New Identity' dialog box contains the following fields and options:

- ID Name***: emma_id
- Participant***: resource:org.acme.vehicle.auction.Member#emma
- ☐ Allow this ID to issue new IDs (👤)
- Text: Issuing an identity generates a one-time secret. You can choose to send this to somebody or use it yourself when it has been issued.
- Buttons**: Cancel, Create New

The 'Identity Issued' dialog box displays the following information:

- Option 1**: Send someone the new user ID and secret so that they can add it to their wallet
 - User ID**: emma_id
 - User Secret**: fcb8ec88
- Option 2**: Use it yourself
 - + Add to my Wallet
- Footer**: ⚠ For security, this secret will only ever be shown once. **Ok**

```
businessNetworkConnection.connect  
('hlfv1', 'my-network', 'emma_id', 'fcb8ec88')
```

Systems of Record Integration

- Domain specific APIs very attractive to mobile and web developers. Resources and operations are business-meaningful
- Composer exploits Loopback framework to create REST APIs: <https://loopback.io/>
- Extensive test facilities for REST methods using loopback
- Composer provides back-end integration with any loopback compatible product
 - e.g. IBM Integration Bus, API Connect, StrongLoop
 - Outbound and Inbound (where supported by middleware)

angular-app

Auctioneer : A participant named Auctioneer

Show/Hide | List Operations | E

CloseBidding : A transaction named CloseBidding

Show/Hide | List Operations | E

Member : A participant named Member

Show/Hide | List Operations | E

Offer : A transaction named Offer

Show/Hide | List Operations | E

Vehicle : An asset named Vehicle

Show/Hide | List Operations | E

GET /Vehicle

Find all instances of the model matched by filter from

POST /Vehicle

Create a new instance of the model and persist it in

GET /Vehicle/{id}

Find a model instance by {{id}} from

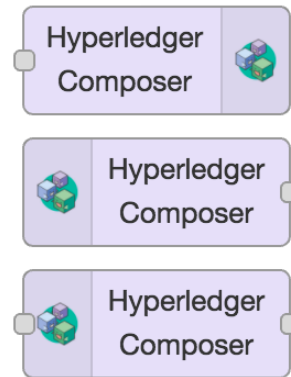
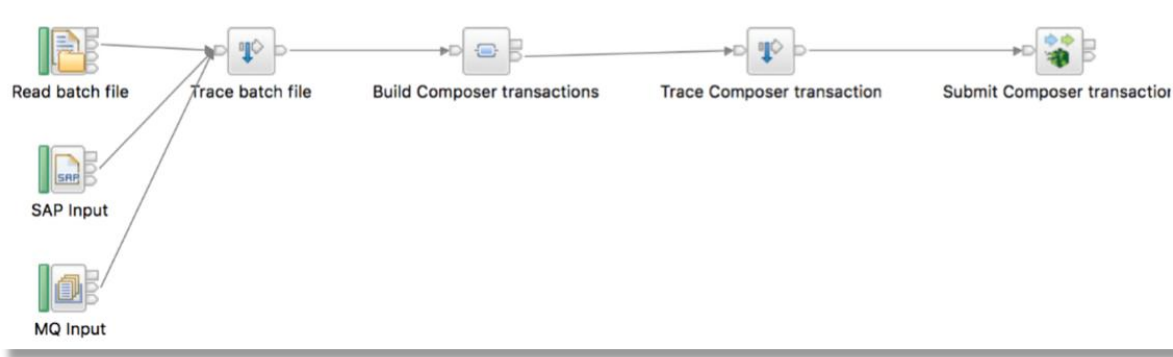
Request URL

http://0.0.0.0:3000/api/Vehicle

Response Body

```
[
  {
    "$class": "org.acme.vehicle.auction.Vehicle",
    "vin": "VEH:1234",
    "owner": "odowda@uk.ibm.com"
  }
]
```


Exploiting Loopback: Examples



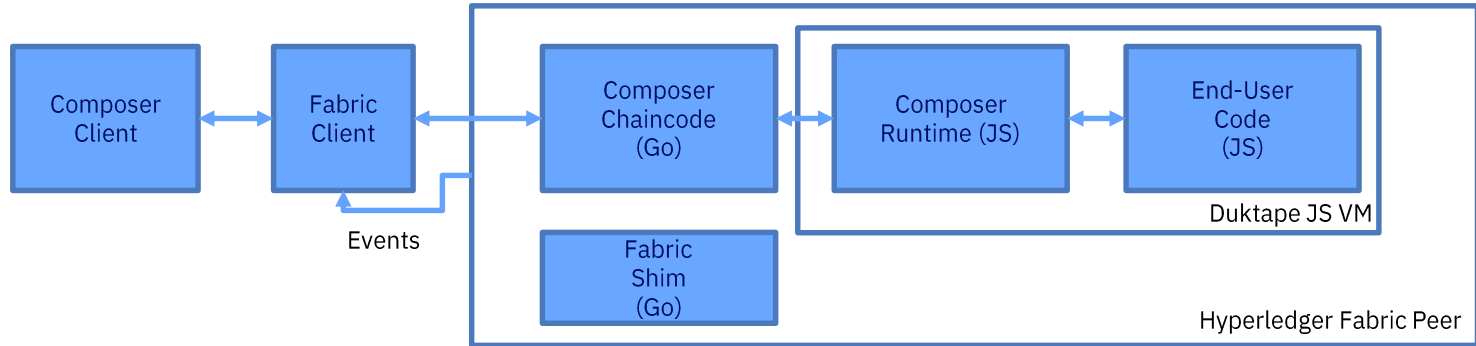
– IBM Integration Bus

- IIB V10 contains Loopback connector
- Example above takes input from file, SAP or MQ
- Data mapping from CSV, BAPI/IDOC or binary form to JSON model definition

– Node.RED

- Pre-built nodes available for Composer
- Connect to hardware devices, APIs and online services
- Install direct from Node.RED UI
 - Manage Palette -> Install -> node-red-contrib-composer

How Composer Maps to Fabric Chaincode



- Each Business Network is deployed to its own chaincode container
 - Container contains a static piece of Go chaincode that starts a Javascript virtual machine running transaction processors
- Browse these containers to view diagnostic information (docker logs)
- Embedded chaincode is not a Composer external interface

Hyperledger Composer Outlook

- Still early in product lifecycle
- Lots of improvements planned
 - See <https://github.com/hyperledger/composer/issues>
- An active development community
 - Open community calls every two weeks
 - Rocket Chat
 - Stack Overflow
- Get involved!

Hyperledger Rocket.Chat

You will need a [Linux Foundation ID](#) , or alternatively you can log in with Facebook, GitHub, Google, or OpenStack.

Let's chat

Stack Overflow

Ask questions in Stack Overflow with the tag [#hyperledger-composer](#).

Ask now

Contribute to the Project

GitHub

Check out the code, feel free to get involved.

GitHub

Community Call

Join our weekly open community calls.

Learn how

Get started with Hyperledger Composer!

- Define, Test and Deploy Business Networks
- Create domain APIs and sample applications
- Integrate existing systems and data

<https://hyperledger.github.io/composer/>

<http://composer-playground.mybluemix.net/>



Thank you

IBM Blockchain

www.ibm.com/blockchain

developer.ibm.com/blockchain

www.hyperledger.org

© Copyright IBM Corporation 2017. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. Any statement of direction represents IBM's current intent, is subject to change or withdrawal, and represents only goals and objectives. IBM, the IBM logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.

