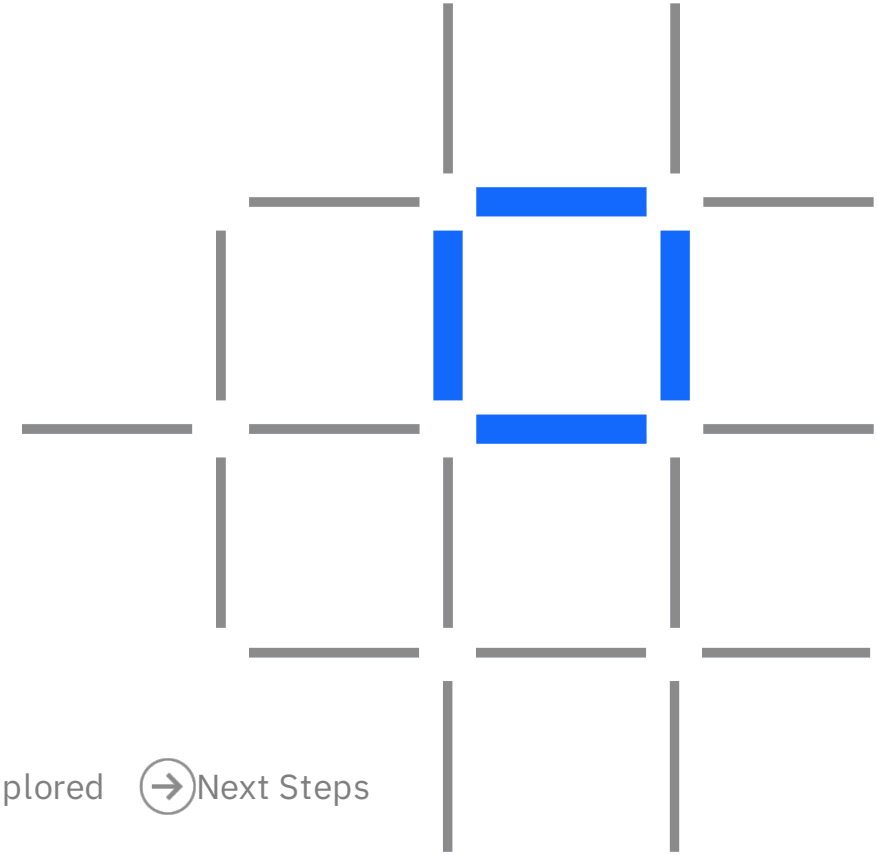
Blockchain Composed

A Technical Introduction to Hyperledger Composer

IBM **Blockchain**



Blockchain education series

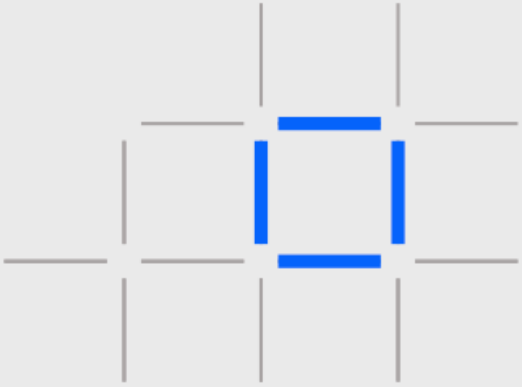
Explained Solutions **Composed** Architected Explored Next Steps

V2.01, 12 October 2017

© 2017 IBM Corporation

IBM

Contents



What is Hyperledger Composer

Application Development

Writing the application

Modeling the business network

Effective Administration

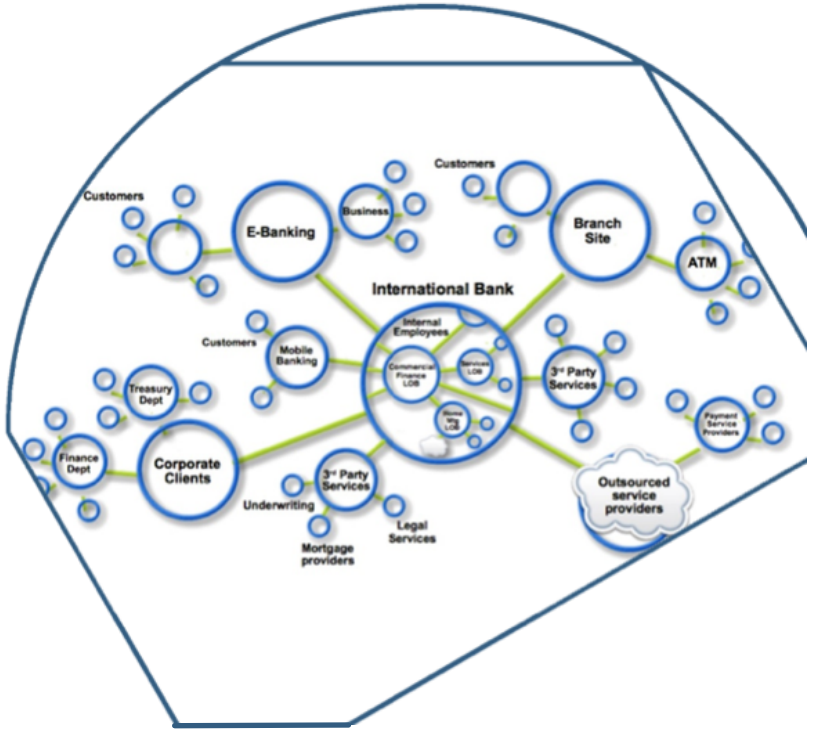
Deploying to a blockchain

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

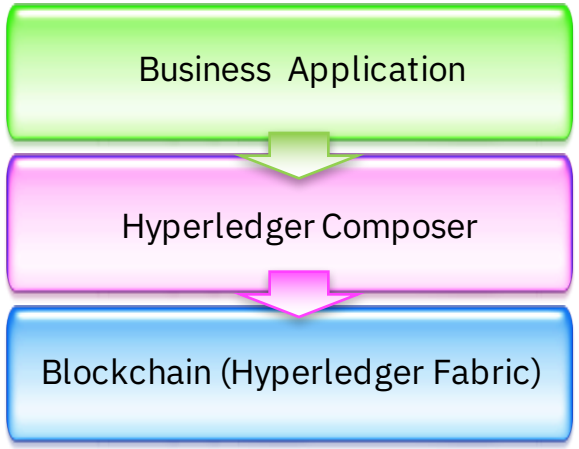


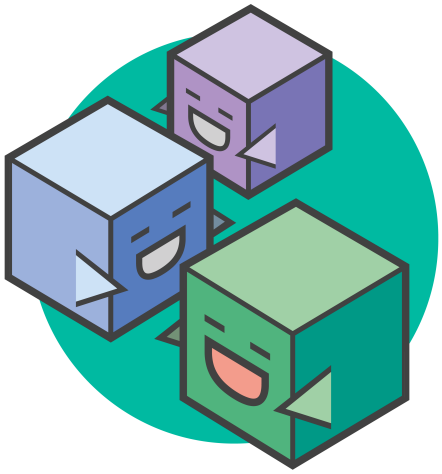
© 2017 IBM Corporation

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/

© 2017 IBM Corporation

Benefits of Hyperledger Composer

Increases understanding

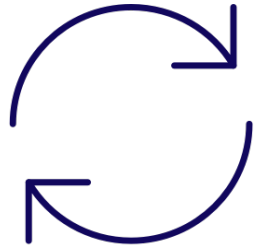
Bridges simply from business concepts to blockchain

Saves time

Develop blockchain applications more quickly and cheaply

Reduces risk

Well tested, efficient design conforms to best practice

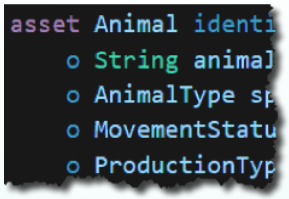


Increases flexibility

Higher level abstraction makes it easier to iterate

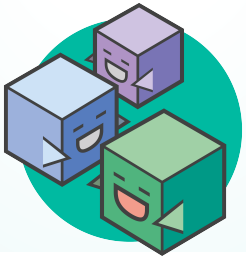
© 2017 IBM Corporation

Extensive, Familiar, Open Development Toolset



Data modelling

JavaScript business logic



Web playground

Client libraries

Editor support

CLI utilities

Code generation

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

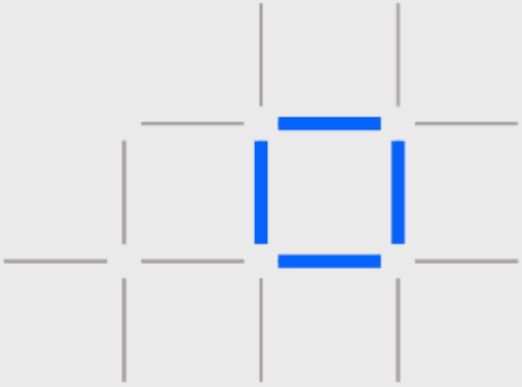
- Provisioning the target environment
- Deploying the business application
- Managing the blockchain

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

© 2017 IBM Corporation

Contents

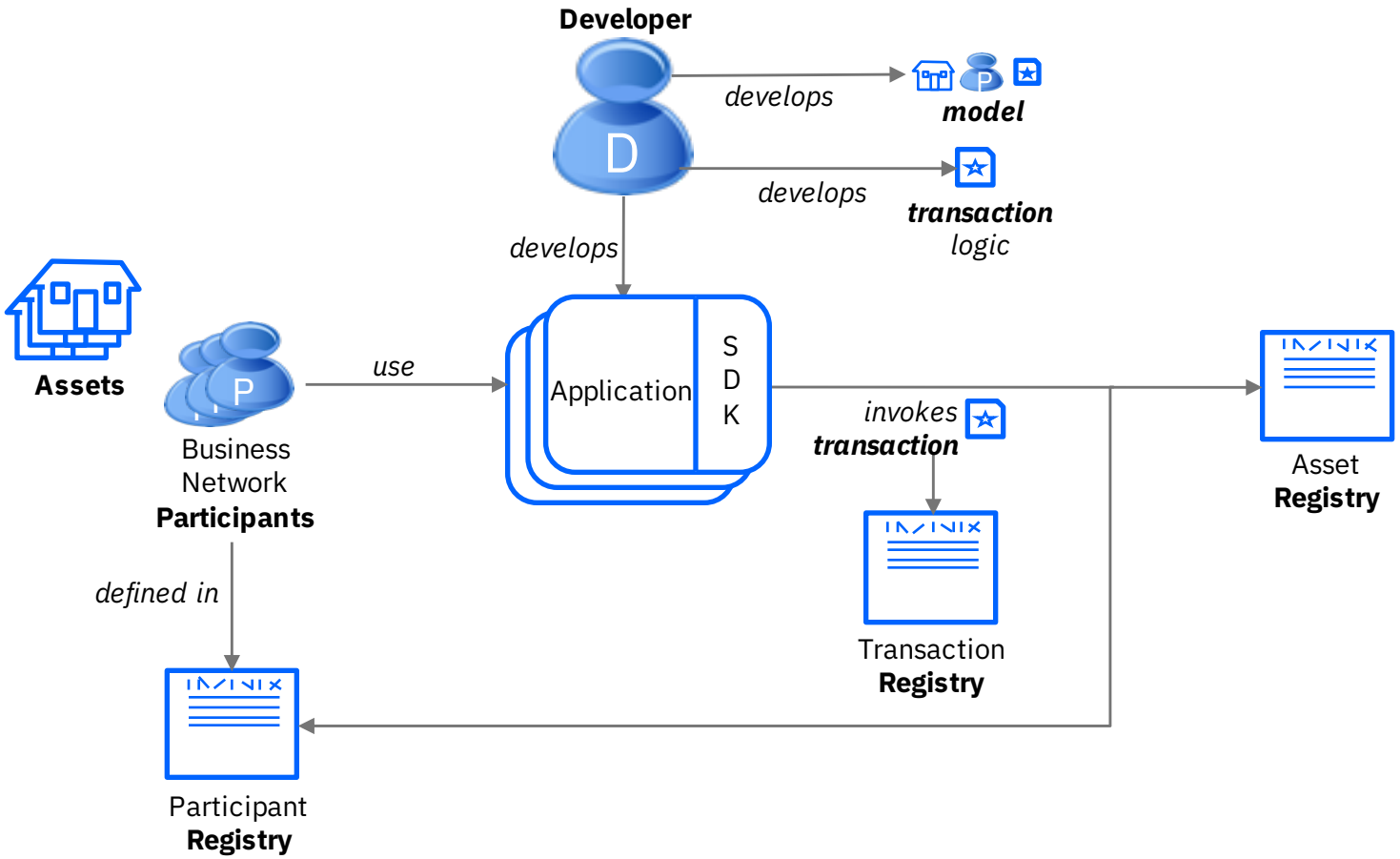


What is Hyperledger Composer

Application Development
Writing the application
Modeling the business network

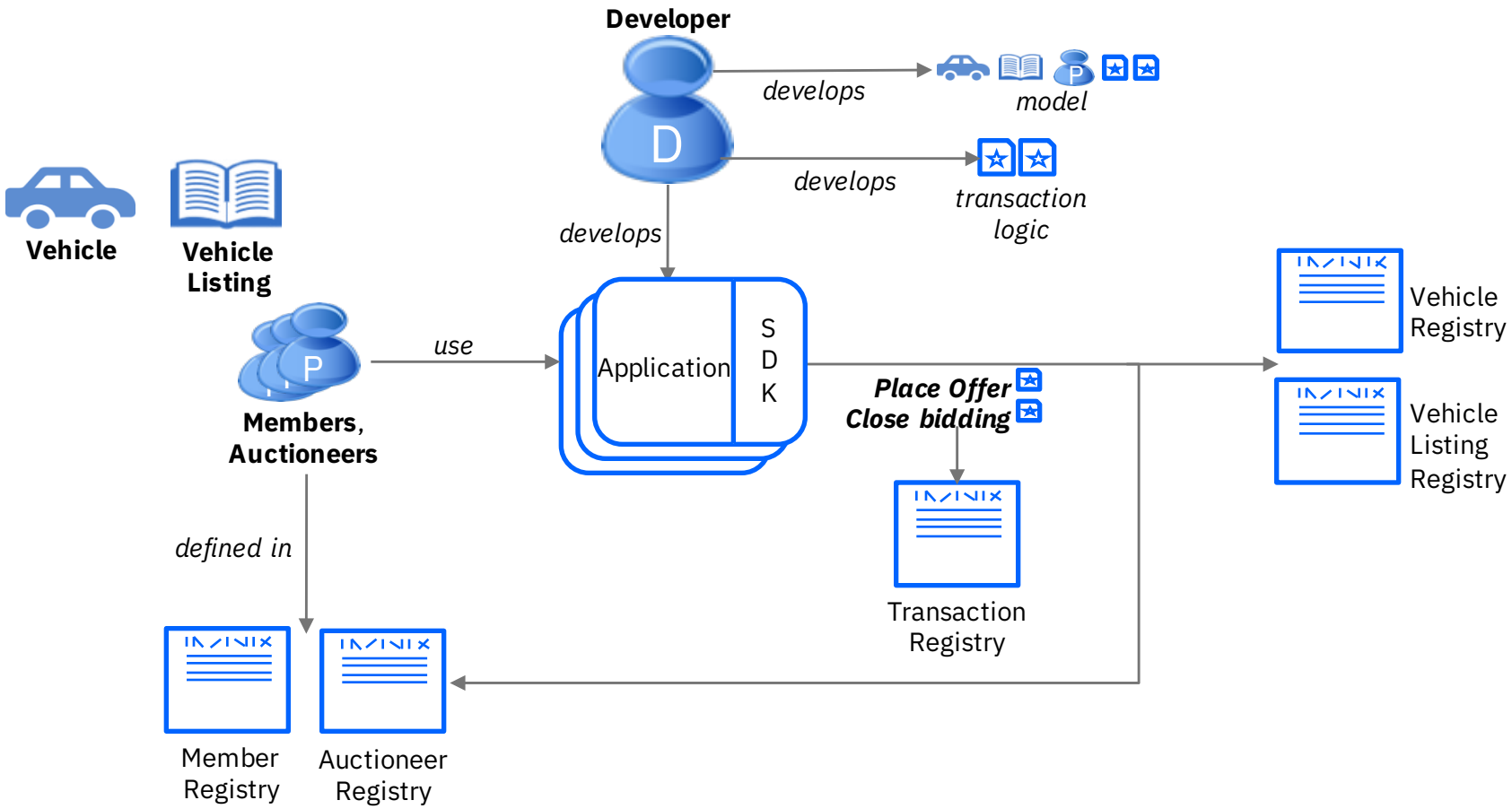
Effective Administration
Deploying to a blockchain
Interacting with systems of record

Key Concepts for the Application Developer



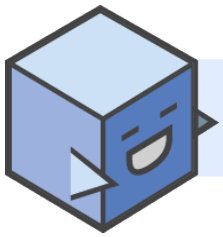
© 2017 IBM Corporation

Example: Vehicle Auction



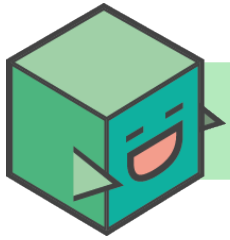
© 2017 IBM Corporation

Developer Concepts



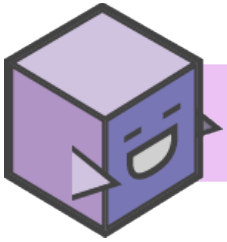
Applications

- 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

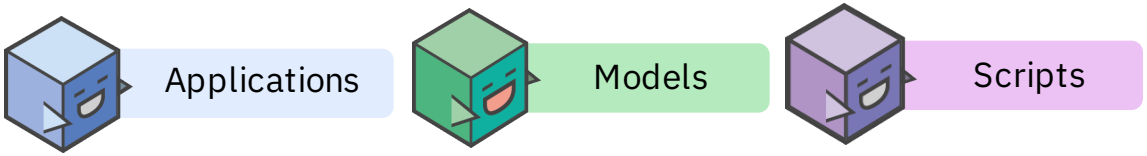


Scripts

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

© 2017 IBM Corporation

Vehicle Auction



Vehicle
VIN
(References a) Member

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

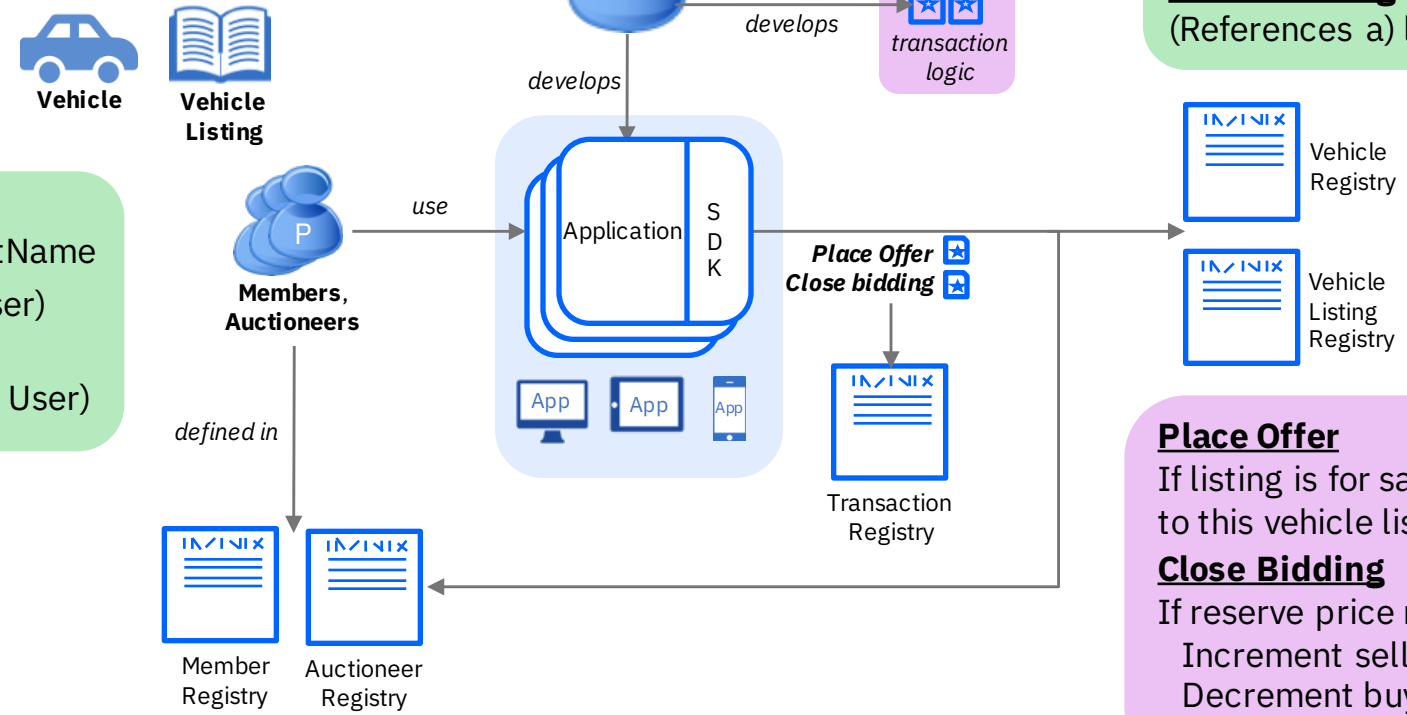
User
email, firstName, lastName

Member (extends User)
balance

Auctioneer (extends User)

Place Offer
bidPrice
(References a) listing, member

Close Bidding
(References a) listing

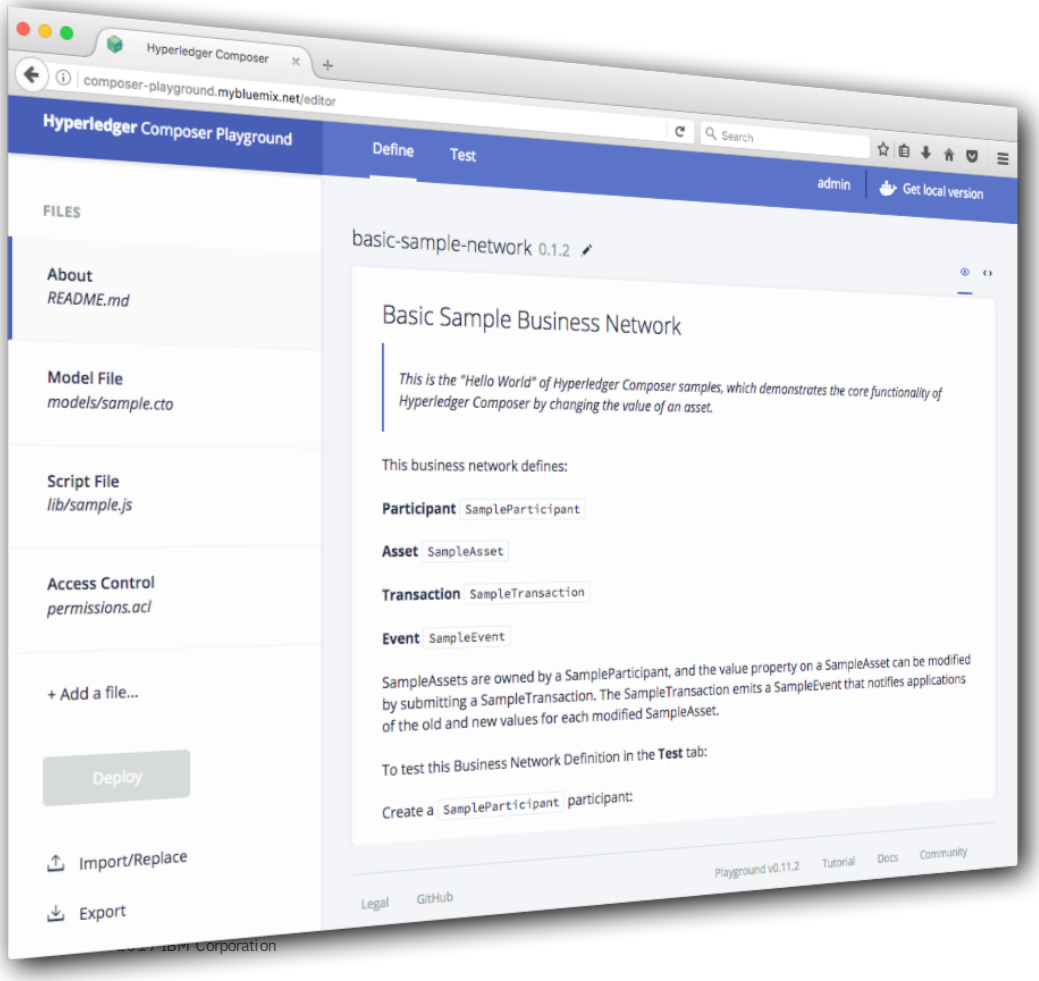


Place Offer
If listing is for sale, add offer to this vehicle listing's offers[]

Close Bidding
If reserve price met
- Increment seller's balance
- Decrement buyer's balance
- Change owner to buyer

© 2017 IBM Corporation

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

© 2017 IBM Corporation

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

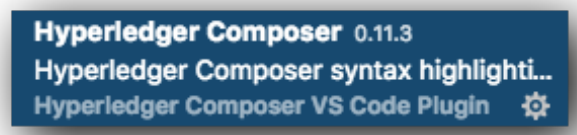
- Install directly from Code Marketplace

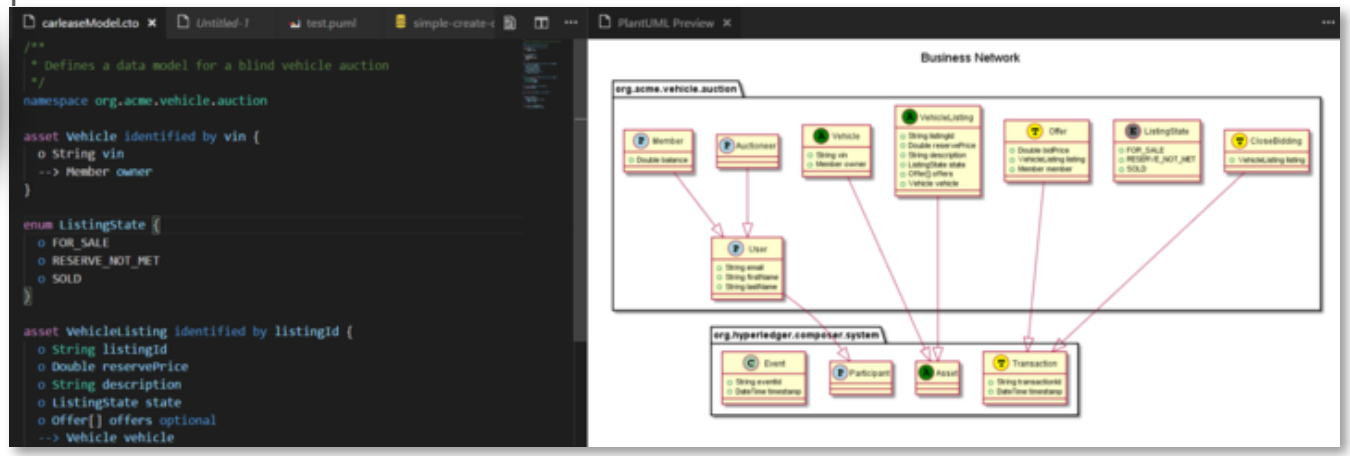
```
namespace org.acme.vehicle.lifecycle

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 Pea
rson
participant PrivateOwner identified by email extends Pearson {
  o String email
}
```





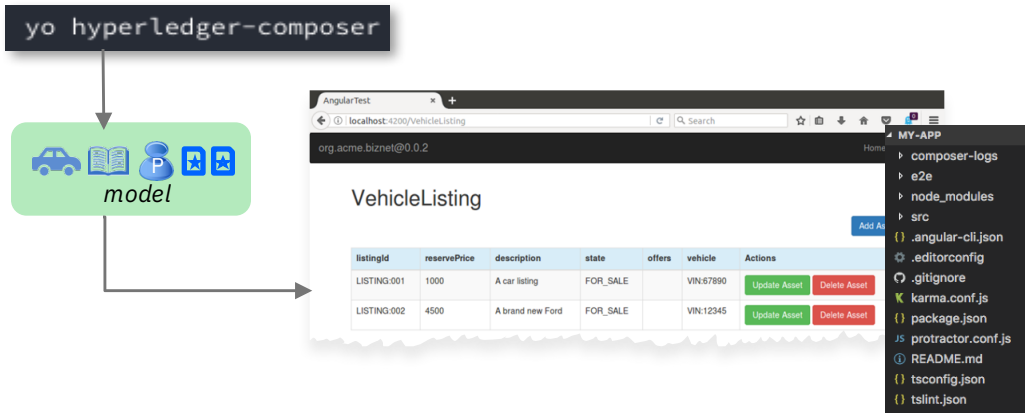
© 2017 IBM Corporation

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
- Also helps with the generation of 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);
```



© 2017 IBM Corporation

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

```
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);
}
```

- 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

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

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

© 2017 IBM Corporation

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
}
```

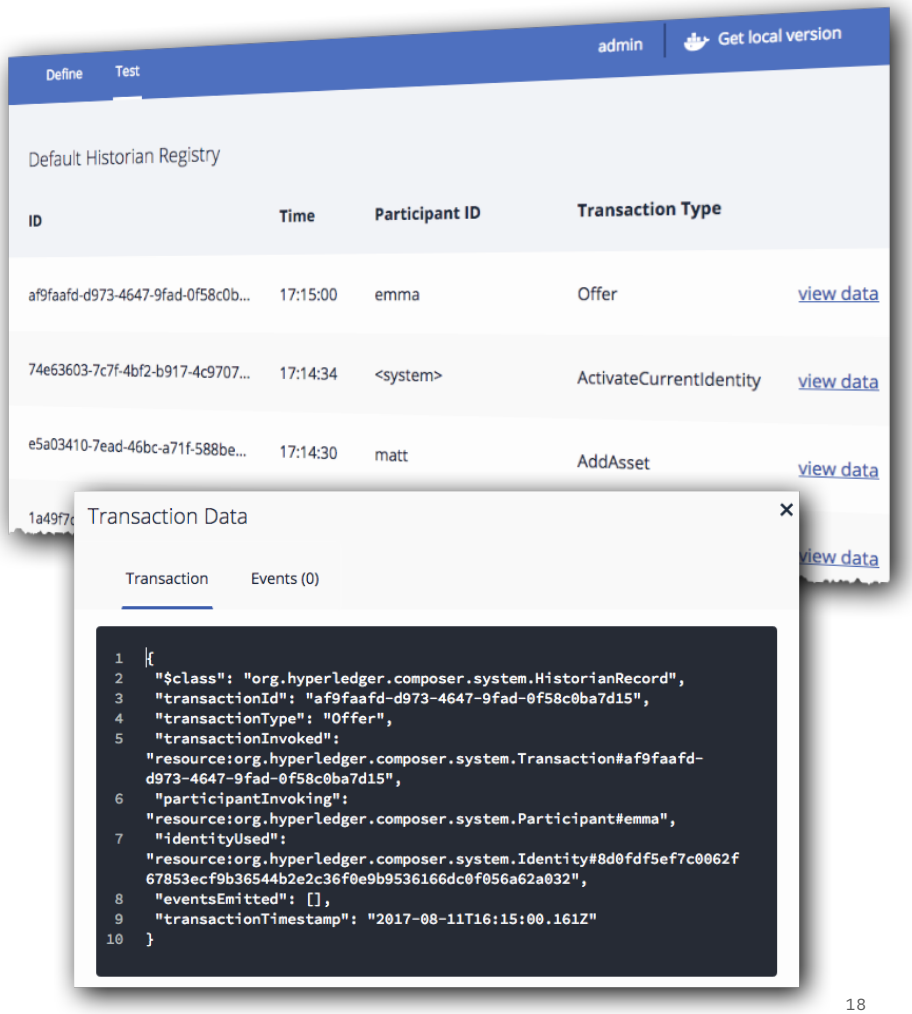
```
rule SystemACL {
  description:  "System ACL to permit all access"
  participant: "org.hyperledger.composer.system.Participant"
  operation: ALL
  resource: "org.hyperledger.composer.system.**"
  action: ALLOW
}
```

- 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

© 2017 IBM Corporation

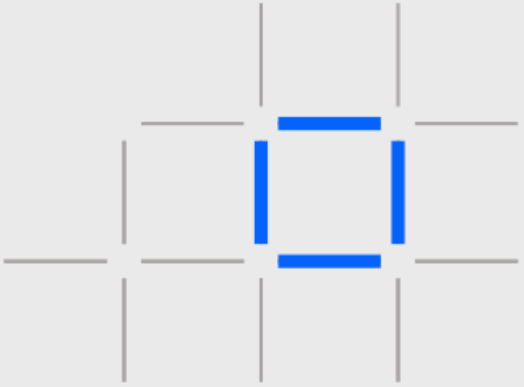
Debugging

- 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



© 2017 IBM Corporation

Contents



What is Hyperledger Composer

Application Development

Writing the application

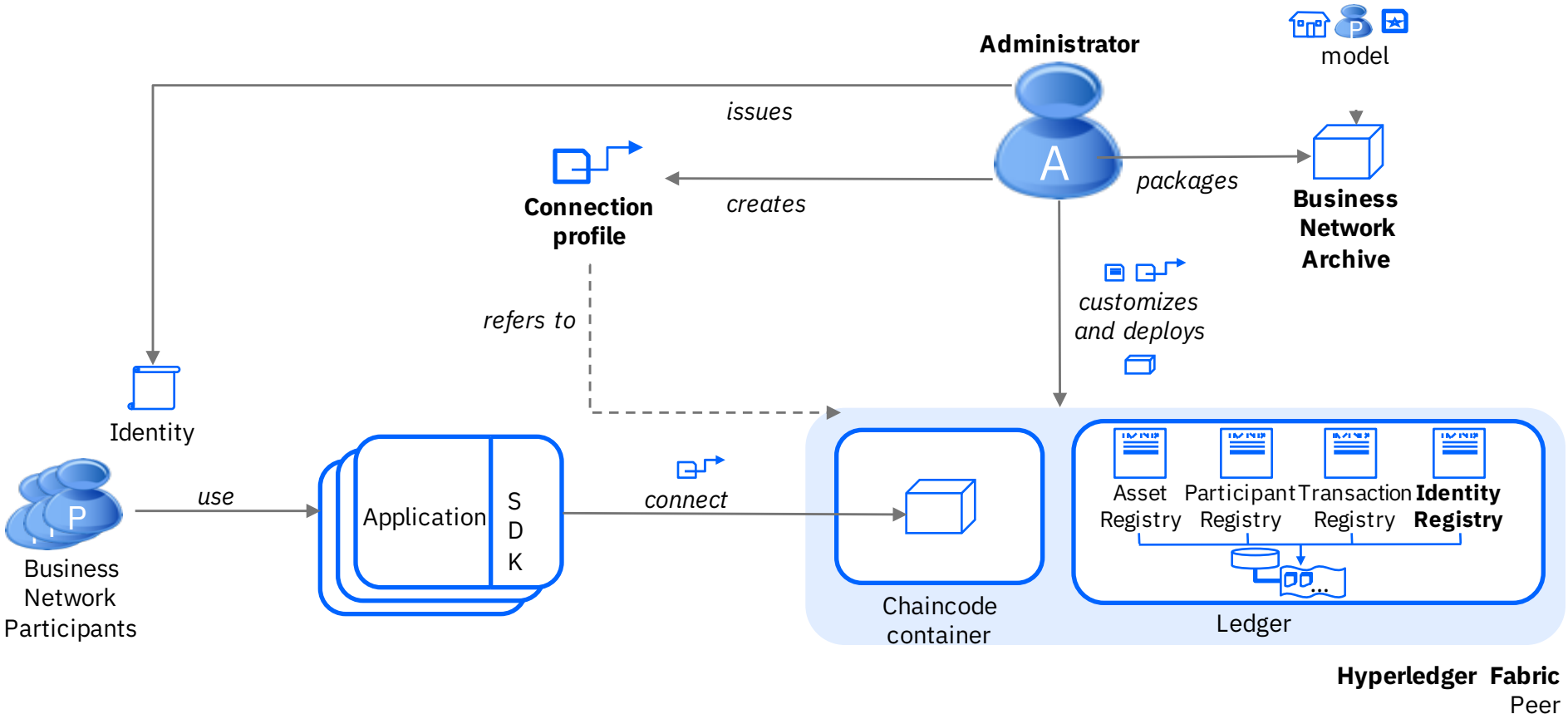
Modeling the business network

Effective Administration

Deploying to a blockchain

Interacting with systems of record

Key Concepts for the Administrator



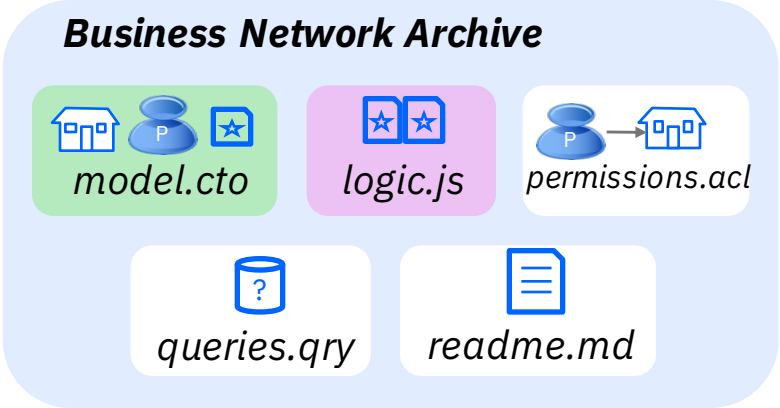
© 2017 IBM Corporation

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…

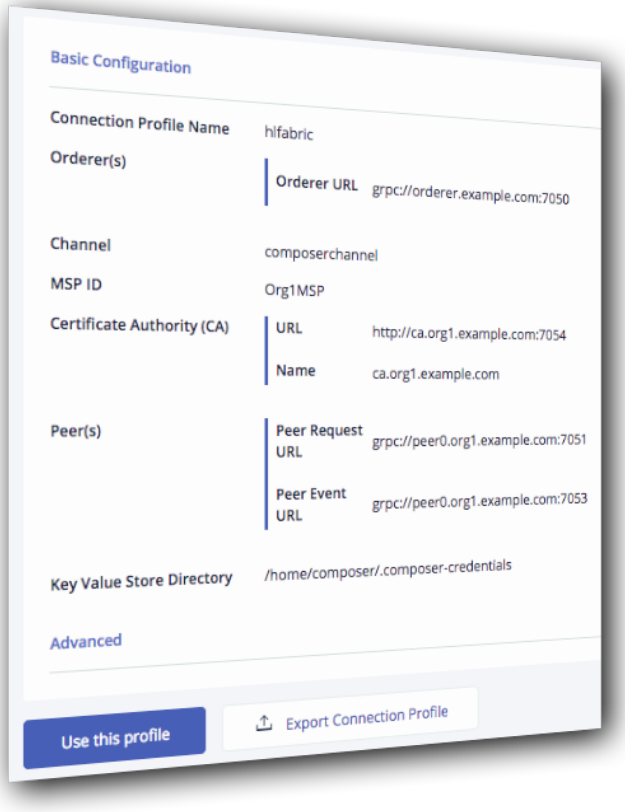
- Create BNA files from Playground or command line
  - Build from filesystem or NPM module




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

© 2017 IBM Corporation

Deployment to Hyperledger Fabric



- 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…

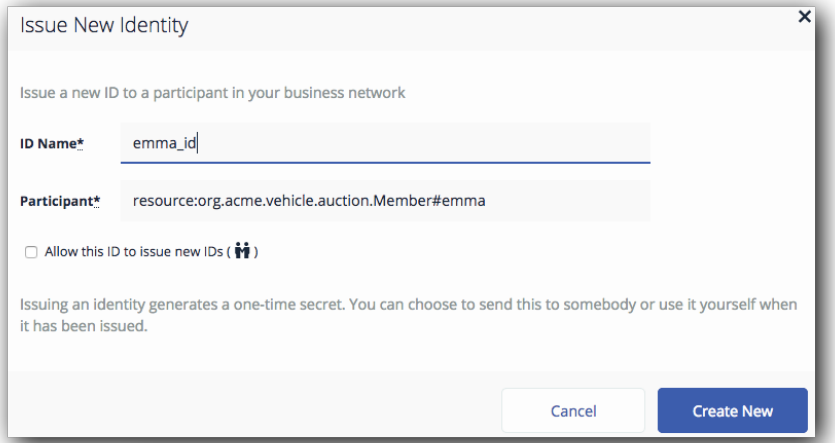
© 2017 IBM Corporation

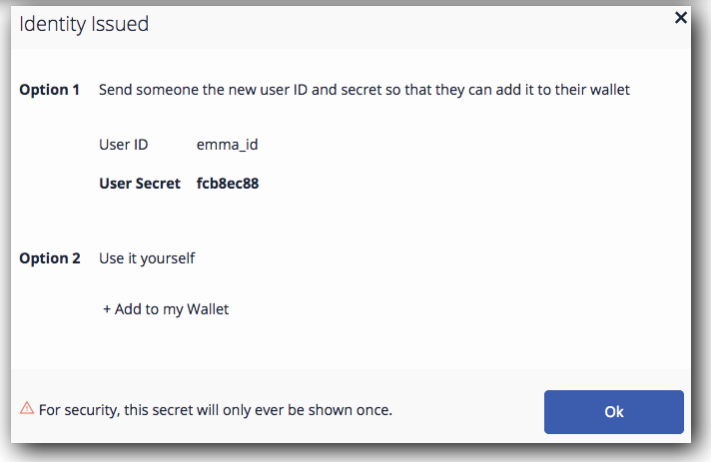
Participant Identity

- 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



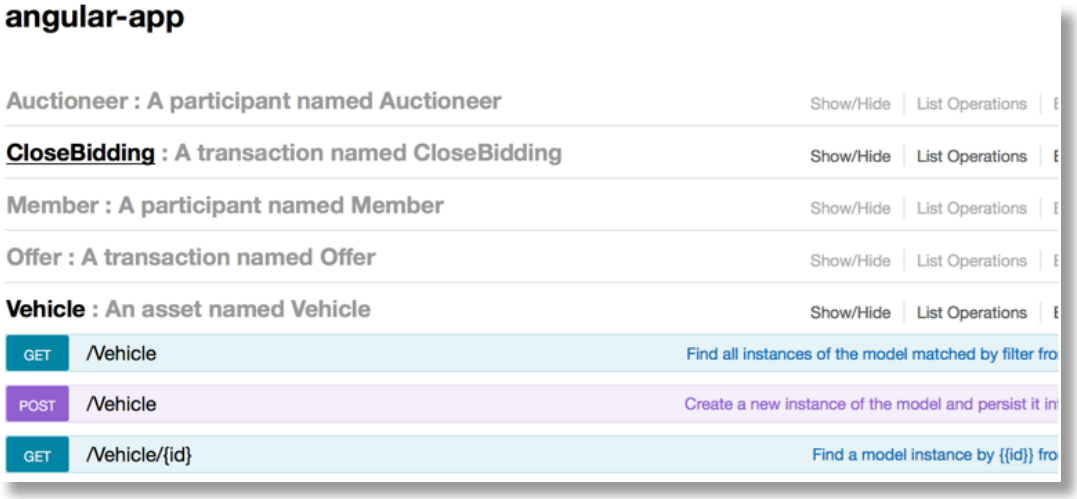



```
businessNetworkConnection.connect
      ('hlfv1', 'my-network', 'emma_id', 'fcb8ec88')
```

© 2017 IBM Corporation

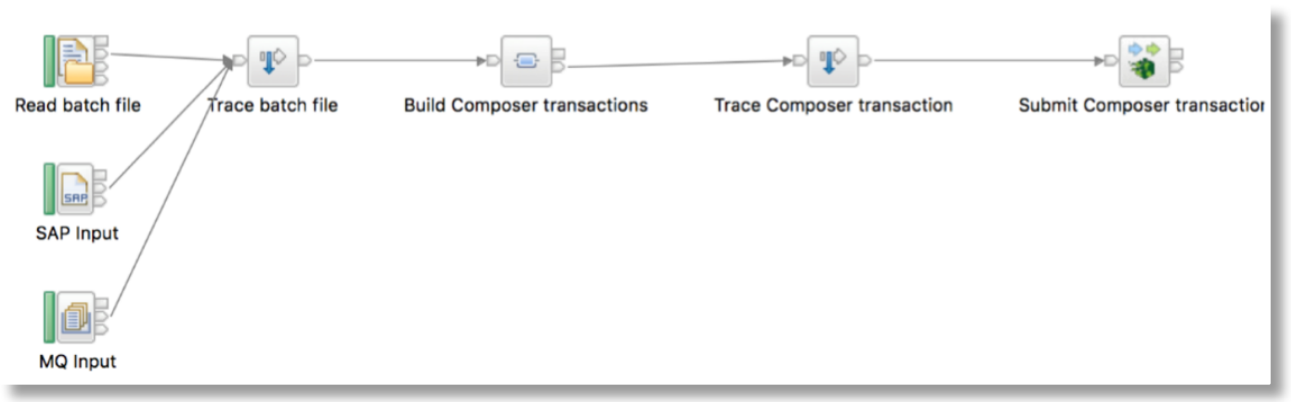
Systems of Record Integration

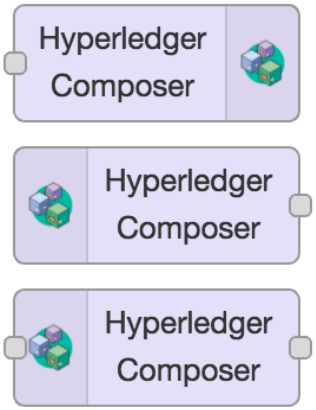
- Domain specific APIs very attractive to mobile and web developers. Resources and operations are business-meaningful
- Composer exploits Loopback framework to create REST APIs: https://loopback.io/
- Extensive test facilities for REST methods using loopback
- Secured using JS Passport, giving >400 options for authentication
- 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)



© 2017 IBM Corporation

Exploiting Loopback: Examples



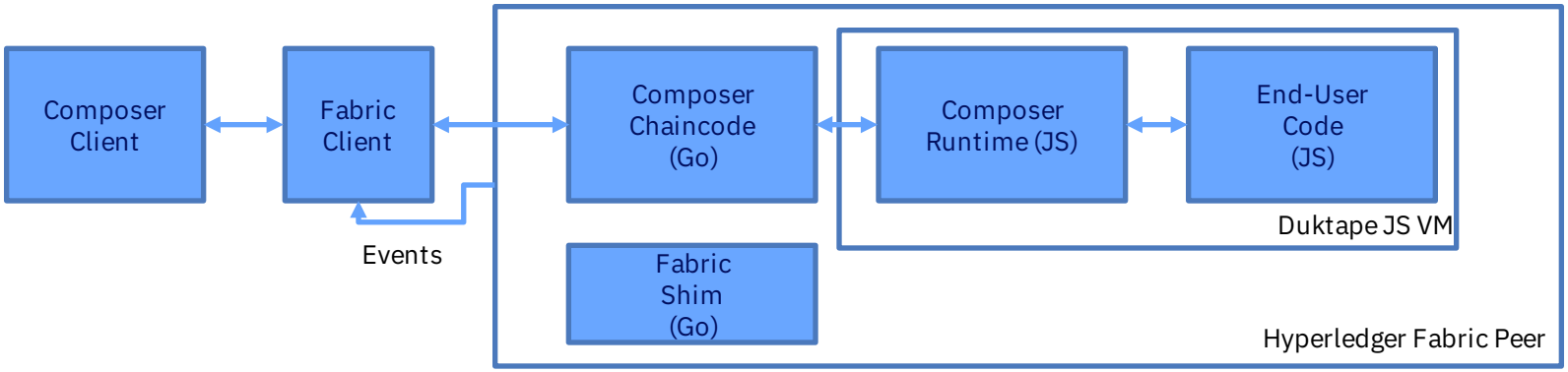


- **IBM Integration Bus**
  - IIB V10 contains Loopback connector
  - Example above takes input from file, SAP or MQ
  - Data mapping from CSV, BAPI/IDOC or binary form to JSON model definition

- **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

© 2017 IBM Corporation

How Composer Maps to Fabric Chaincode

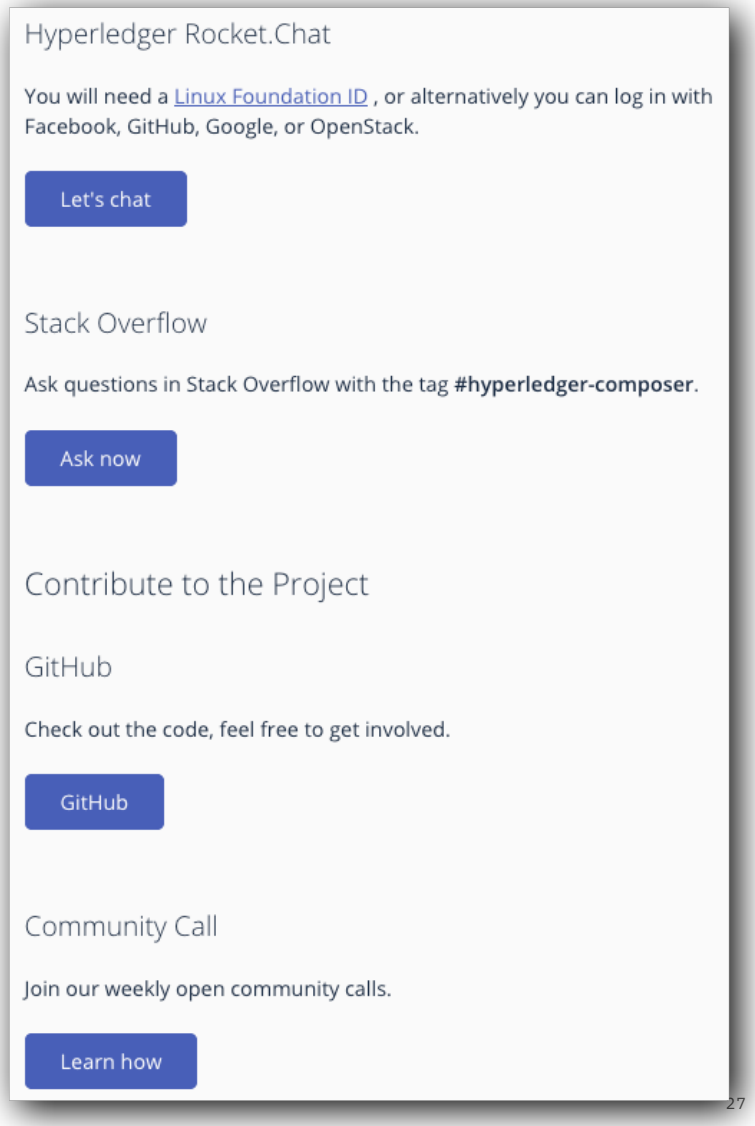


- Each Business Network is deployed to its own chaincode container
  - Container contains a static piece of Go chaincode that starts a Javascript virtual machine running transaction processors
- Browse these containers to view diagnostic information (docker logs)
- Embedded chaincode is not a Composer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

© 2017 IBM Corporation

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!



© 2017 IBM Corporation

Get started with Hyperledger Composer!

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

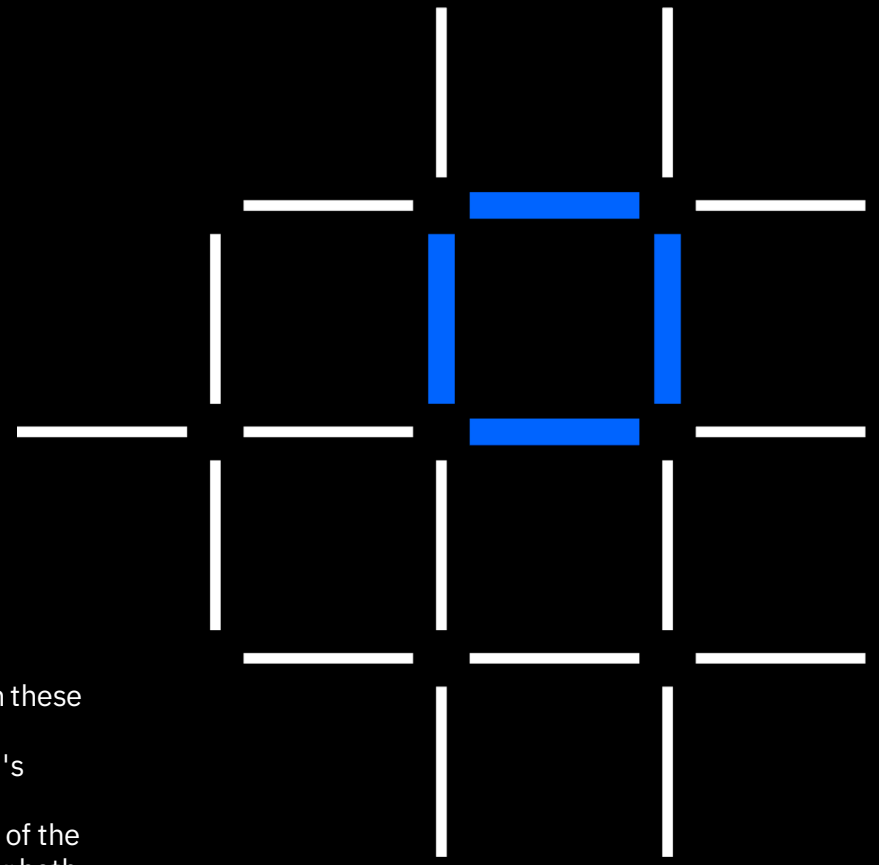
https://hyperledger.github.io/composer/

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

Thank you

Jennifer Foley
foleyje@us.ibm.com
Integration Architect
WW Client Center for System Innovation

IBM **Blockchain**



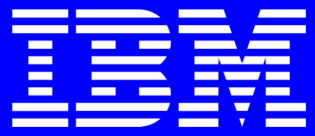
www.ibm.com/blockchain

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.

IBM


IBM