IBM Blockchain

IBM Blockchain Hands-On
Blockchain Fabric Composer

Lab One – Exercises

Contents

Overview

The purpose of this lab is to introduce you to the concepts of a blockchain by showing you how a blockchain transfers assets between participants in a business network. We will use car leasing as the scenario for the demo.

Introduction

Pre-requisites:

- Internet Connection
- Any Modern Browser (IE9+)

No login is required to access the demo site.

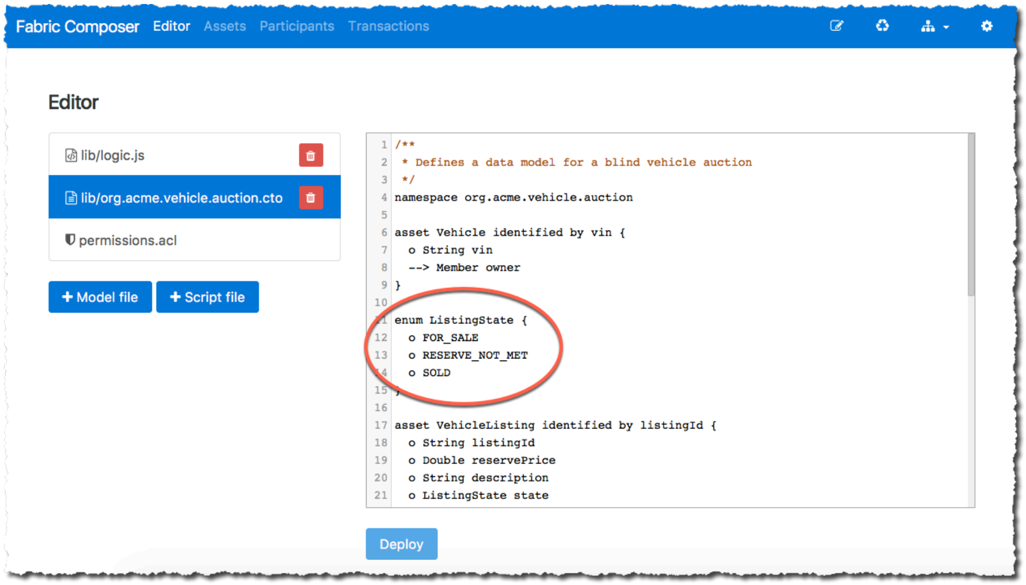
The demo site simulates the entire blockchain network within the browser providing a sandpit environment to test and explore the capabilities of Fabric Composer UI. It is then possible to connect this to a live blockchain Hyperledger Fabric instance on the local machine or install the Fabric Composer on a local machine for more developer friendly tools

Fabric Composer-UI is one method to use Fabric Composer, other methods are also available at www.fabric-composer.org.

Section 1. Using Fabric Composer UI

1.1. Open the Fabric Composer UI

__1. Go to fabric-composer.mybluemix.net



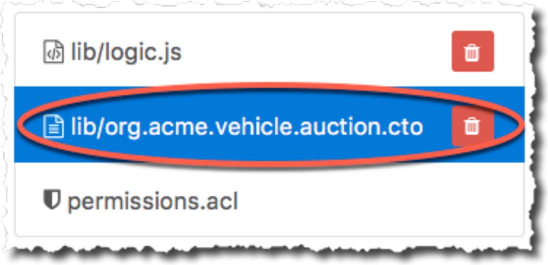
```
/**
 * Defines a data model for a blind vehicle auction
 */
namespace org.acme.vehicle.auction

asset Vehicle identified by vin {
  o String vin
  --> Member owner
}

enum ListingState {
  o FOR_SALE
  o RESERVE_NOT_MET
  o SOLD
}

asset VehicleListing identified by listingId {
  o String listingId
  o Double reservePrice
  o String description
  o ListingState state
```

__2. Ensure the demo loaded is the Car Auction Example showing easily identified by the FOR_SALE, RESERVE_NOT_MET… items on the page shown above. If this is not present see Appendix A to reset demo.

1.2. Explore the Editor Views

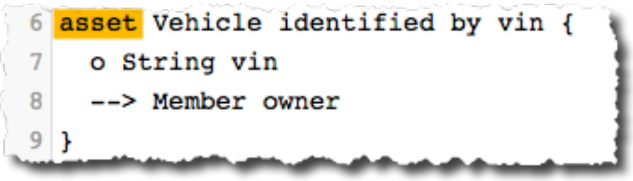
1.2.1. Model File

__3. Go to the Model File if it is not already open



This .cto file defines the namespace for assets, participants and transactions are modeled

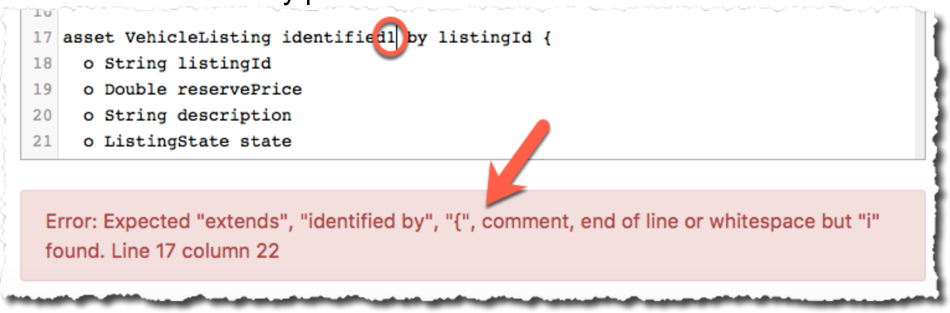
__4. Look at the asset defined



```
asset Vehicle identified by vin {
  o String vin
  --> Member owner
}
```

This uses the Fabric Composer Modeling Language which will be looked at more later. An *asset* can be items such as bonds, mortgages or vehicles in this case. Here we can see the asset is called ‘*Vehicle*’ and will have an associated *vin* and relationship a ‘*Member*’ that we would call an ‘*owner*’.

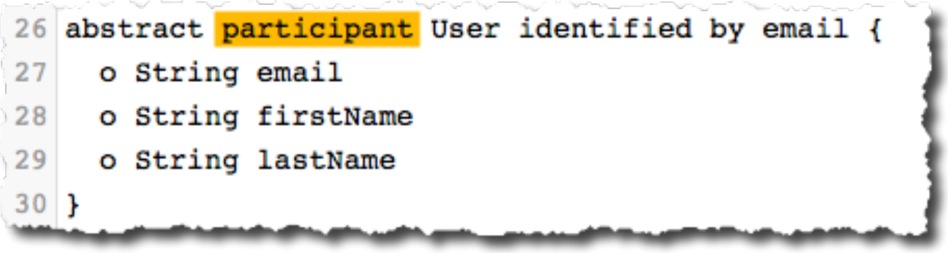
__5. Type & romove a number in at any point to show the live validation



```
17 asset VehicleListing identified1 by listingId {
18   o String listingId
19   o Double reservePrice
20   o String description
21   o ListingState state
```

Error: Expected "extends", "identified by", "{", comment, end of line or whitespace but "i" found. Line 17 column 22

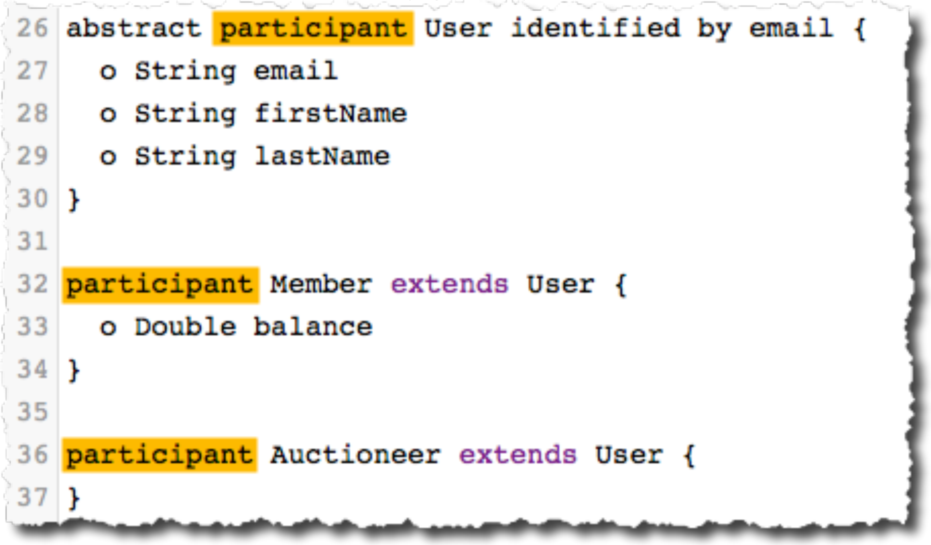
__6. Scroll down to look at the participant namespace defined



```
abstract participant User identified by email {
  o String email
  o String firstName
  o String lastName
}
```

The *participant* will be the people or companies within the business network. This participant called 'User' will be defined as having a *email*, *firstName* and *lastName*. An ***abstract*** shows this declaration cannot be created and would need to be part of one of the extentions below

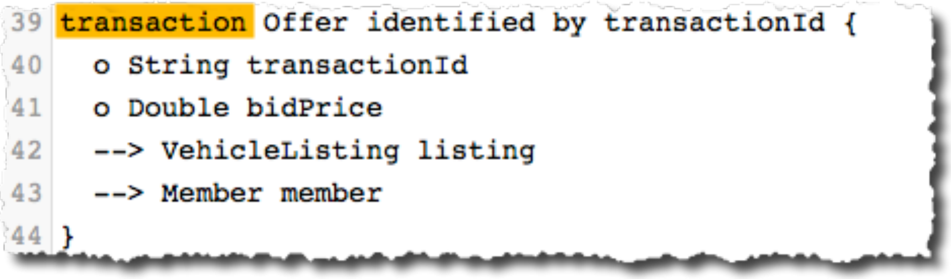
__7. See how a type can be extended to effectivly group types and add more data



```
abstract participant User identified by email {
  o String email
  o String firstName
  o String lastName
}

participant Member extends User {
  o Double balance
}

participant Auctioneer extends User {
}
```

Here the user can become a *Member* requiring a *balance* or an *Auctioneer*

__8. Look at the transaction namespace defined

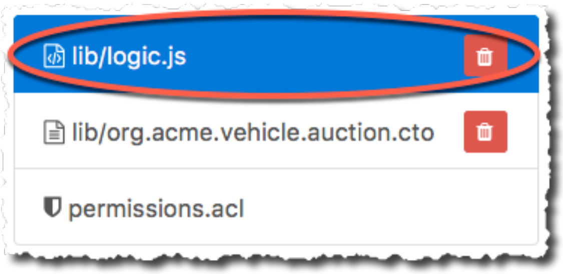


```
transaction Offer identified by transactionId {
  o String transactionId
  o Double bidPrice
  --> VehicleListing listing
  --> Member member
}
```

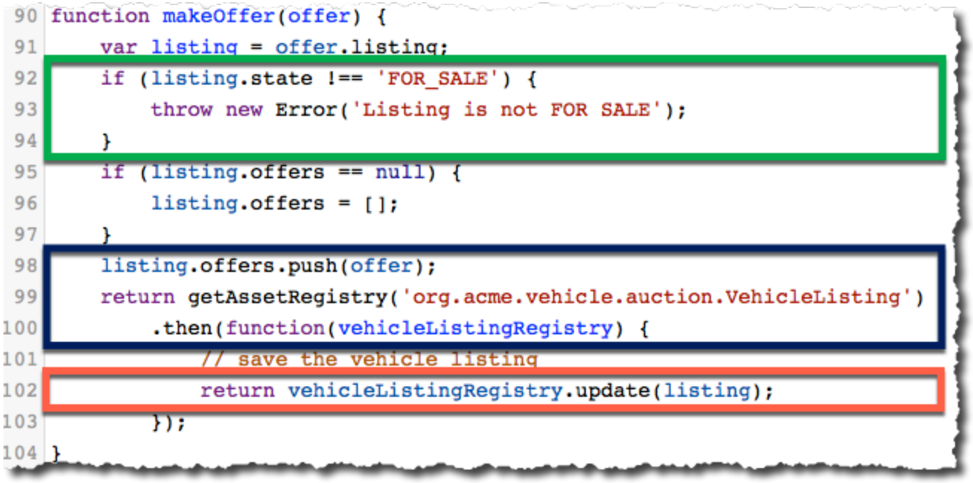
The *transaction's* start to define the business rules and logic required to make a transaction. This section defines what a transaction must have and it's relations to other definititions set out.

1.2.2. Script file

__9. Open the script file



__10. Scroll to the bottom to see the logic used to *makeOffer* on a car being auctioned



```
function makeOffer(offer) {
    var listing = offer.listing;
    if (listing.state !== 'FOR_SALE') {
        throw new Error('Listing is not FOR SALE');
    }
    if (listing.offers == null) {
        listing.offers = [];
    }
    listing.offers.push(offer);
    return getAssetRegistry('org.acme.vehicle.auction.VehicleListing')
        .then(function(vehicleListingRegistry) {
            // save the vehicle listing
            return vehicleListingRegistry.update(listing);
        });
}
```

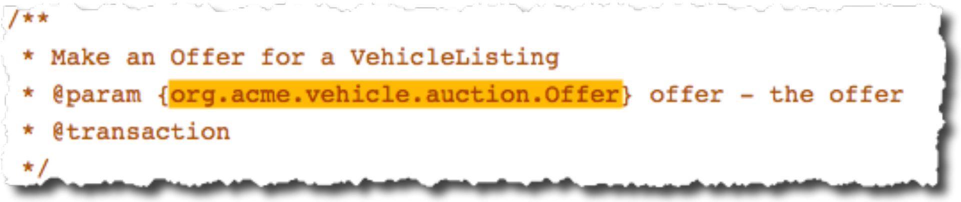
This does is the javascript logic to define when each type of transaction can take place. WE can see the components working here

The logic that the vehicle must be for sale to submit an offer on it

The retrieves the asset registry to then update them a few lines later

Save the updated asset back to the registry

__11. See the annotations above each transaction

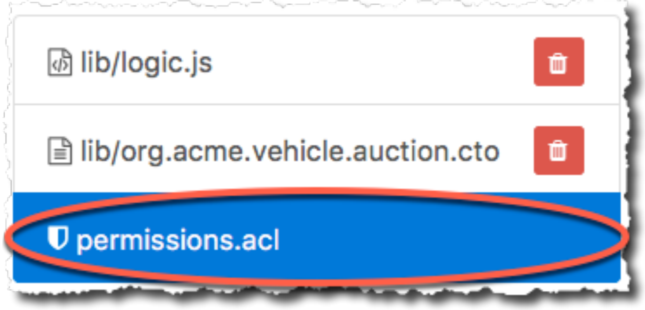


```
/**
 * Make an Offer for a VehicleListing
 * @param {org.acme.vehicle.auction.Offer} offer - the offer
 * @transaction
 */
```

__12. The annotations are a custom tag to link the modeled transaction to the transaction processor. So whenever a org.acme.vehicle.auction.Offer is called it will submit to this logic.

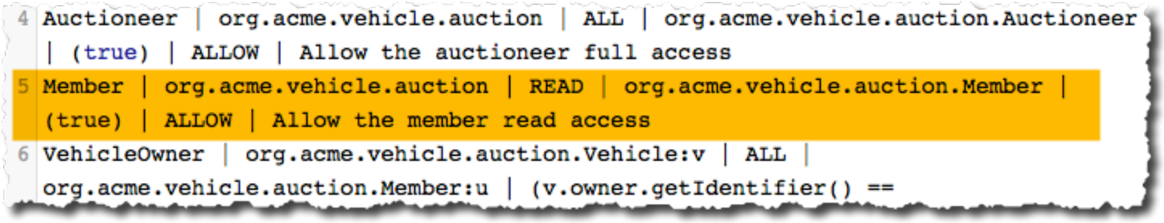
1.2.3. Access Control List

__13. Open the Access Control List (ACL)



The ACL defines who can do what within the network with rules run in order

__14. Look at the ACL rules defined



```
Auctioneer | org.acme.vehicle.auction | ALL | org.acme.vehicle.auction.Auctioneer
| (true) | ALLOW | Allow the auctioneer full access
Member | org.acme.vehicle.auction | READ | org.acme.vehicle.auction.Member |
(true) | ALLOW | Allow the member read access
VehicleOwner | org.acme.vehicle.auction.Vehicle:v | ALL |
org.acme.vehicle.auction.Member:u | (v.owner.getIdentifier() ==
```

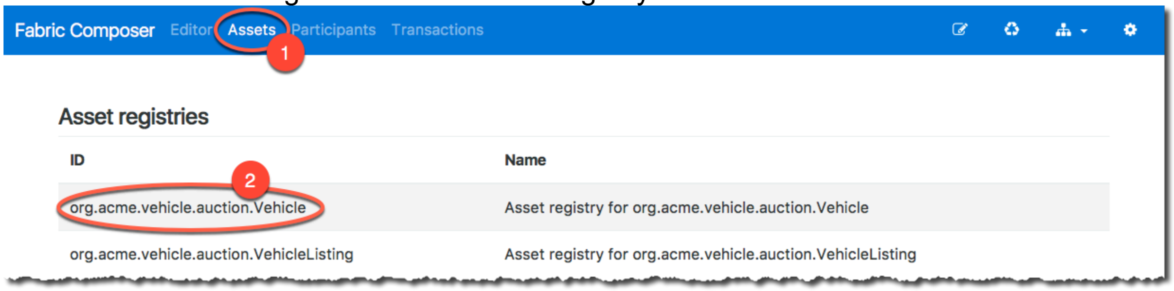
The rule allowes or denys all users to access aspects of the org.acme.vehicle namespace. Rules follow the **ID | NOUN | VERB | PARTICIPANT | PREDICATE | ACTION | DESCRIPTION** format, more on this formatting later. There must be at least 1 ACL rule defined as if no rule is found relating to a specific transaction, the transaction is thrown out.

The highlighted rule shows:

Format Item	Defined as	Meaning
ID	Member	Name of rule is 'Member'
NOUN	org.acme.vehicle.auction	Area of access being defined
VERB	READ	READ access is being allowed or denied
PREDICATE	org.acme.vehicle.auction.Member	Who is being allowed or denied acces
ACTION	(true)	Set of rules to follow, in this case always follow

1.3. Add an Asset

__15. Open the list of Asset Registries and .Vehicle registry

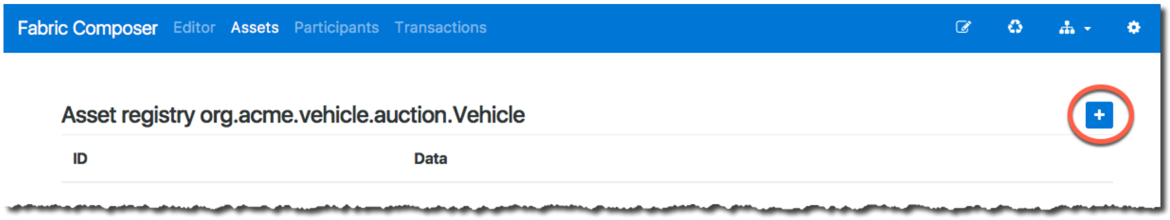


Asset registries

ID	Name
org.acme.vehicle.auction.Vehicle	Asset registry for org.acme.vehicle.auction.Vehicle
org.acme.vehicle.auction.VehicleListing	Asset registry for org.acme.vehicle.auction.VehicleListing

The asset registries view shows the list of each asset registry (of which there is two in this example), these registrys contains all the assets of that type.

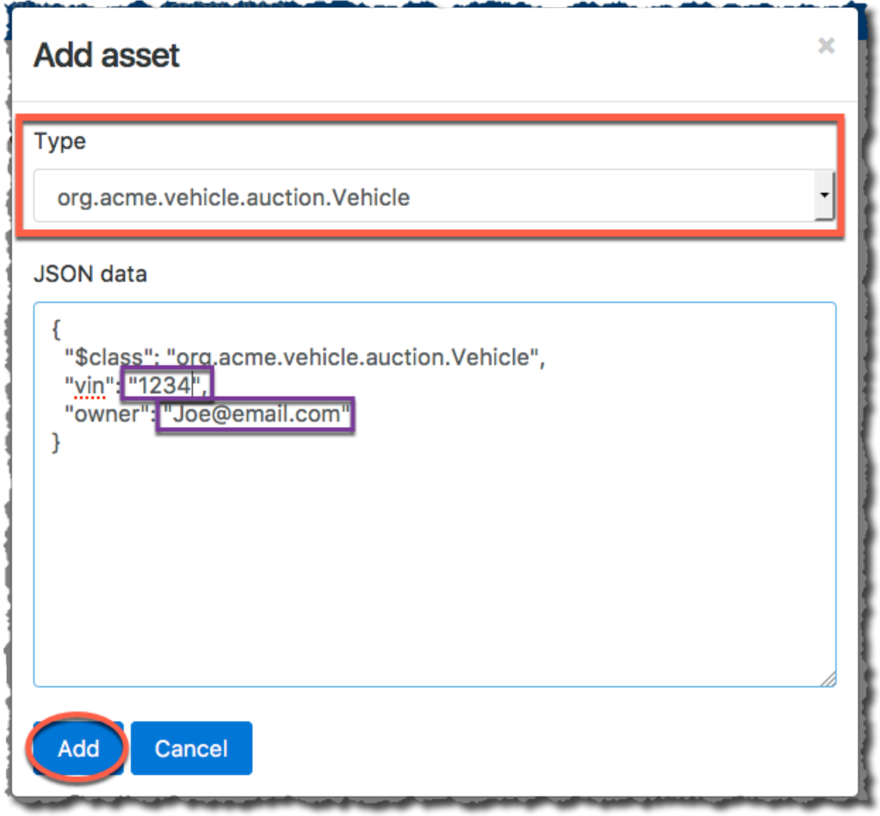
__16. This *org.acme.biznet.auction.vehicle* registry contains no assets already. **Click the '+'** to add a new asset.



Asset registry org.acme.vehicle.auction.Vehicle

ID	Data

__17. The *Add Asset* view then lets you submit the call to add a new asset in JSON structure to input



Add asset

Type

org.acme.vehicle.auction.Vehicle

JSON data

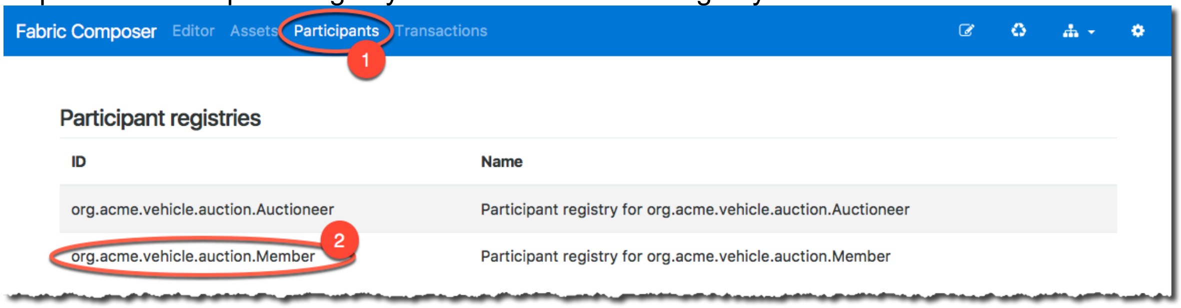
```
{
  "$class": "org.acme.vehicle.auction.Vehicle",
  "vin": "1234",
  "owner": "Joe@email.com"
}
```

Choose the vin number (1234) for the car and an owner then hit 'Add'

__18. View your newly added asset in the registry

1.4. Add a Participant

__19. Open the Participant Registrys view and .Member registry



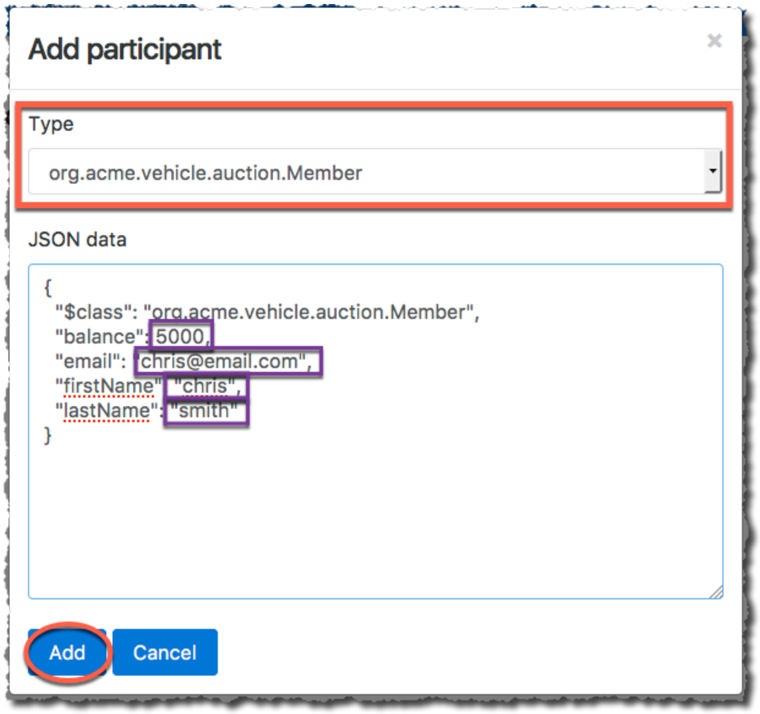
Participant registries

ID	Name
org.acme.vehicle.auction.Auctioneer	Participant registry for org.acme.vehicle.auction.Auctioneer
org.acme.vehicle.auction.Member	Participant registry for org.acme.vehicle.auction.Member

The participant registries view shows the list of each particiapnt registry (of which there are two in this example), these registrys contains all the participants of that type.

__20. Click the + to add a new Member

__21. Add the details to the JSON data structure that will be submitted



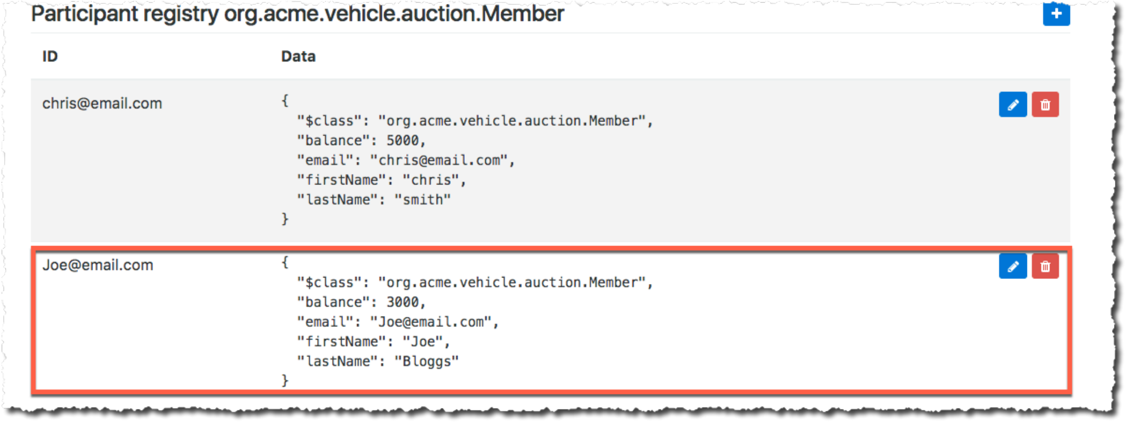
Add participant

Type: org.acme.vehicle.auction.Member

JSON data

```
{
  "$class": "org.acme.vehicle.auction.Member",
  "balance": 5000,
  "email": "chris@email.com",
  "firstName": "chris",
  "lastName": "smith"
}
```

__22. Do the same to add the current owner of the vehicle (Joe@email.com)



Participant registry org.acme.vehicle.auction.Member

ID	Data

chris@email.com	{
	"$class": "org.acme.vehicle.auction.Member",
	"balance": 5000,
	"email": "chris@email.com",
	"firstName": "chris",
	"lastName": "smith"
	}

Joe@email.com	{
	"$class": "org.acme.vehicle.auction.Member",
	"balance": 3000,
	"email": "Joe@email.com",
	"firstName": "Joe",
	"lastName": "Bloggs"
	}

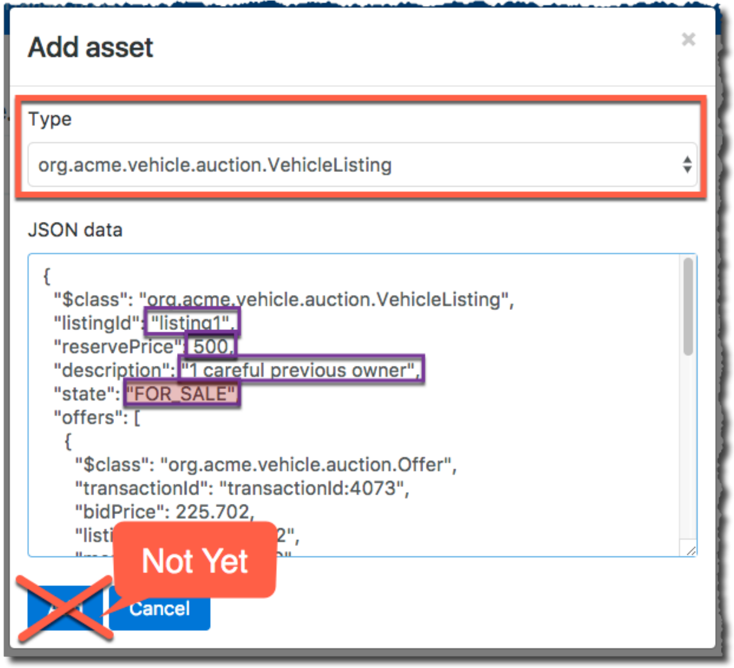
1.5. Add asset to Vehicle Listing

In auctions, the assets being sold are put into a “catalogue” of all assets being sold at auction, this step will add the preveously created vehicle into the VehicleListing which is the equivilant of the catalogue.

__23. Open the o**rg.acme.vehicle.auction.VehicleListing** registry from the asset tab

__24. Click the + icon to add the asset

__25. See the transaction type is aadding the asset to the correct regisrty and then complete the JSON structure to add the asset to the registry. **Ensure the “state” is “FOR_SALE”**



Type

org.acme.vehicle.auction.VehicleListing

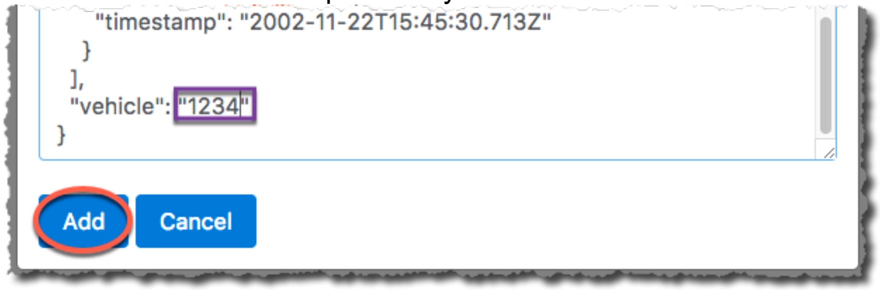
JSON data

```
{
  "$class": "org.acme.vehicle.auction.VehicleListing",
  "listingId": "listing1",
  "reservePrice": 500,
  "description": "1 careful previous owner",
  "state": "FOR_SALE",
  "offers": [
   {
    "$class": "org.acme.vehicle.auction.Offer",
    "transactionId": "transactionId:4073",
    "bidPrice": 225.702,
    "listi                2",
```

Ensure the “state” is “FOR_SALE” or the asset cannot be put up for auction due to the business rules set out.

The information in the middle sets simulates other participants submitting bids on this vehicle

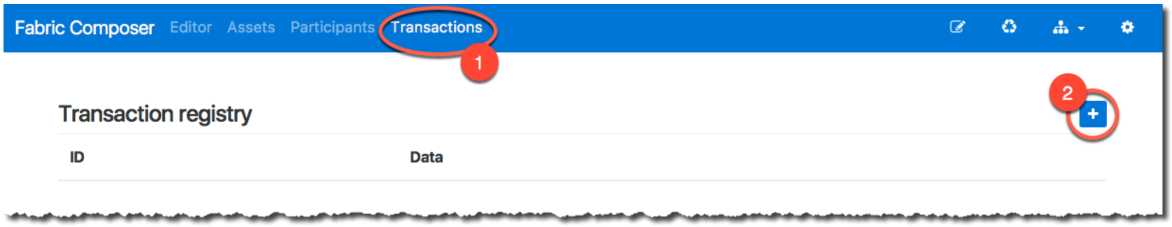
__26. Scroll down and find the *“vehicle”* that should be added to the auction. Add the same *VIN* number of the vehicle that was created previously



```
      "timestamp": "2002-11-22T15:45:30.713Z"
    }
  ],
  "vehicle": "1234"
}
```

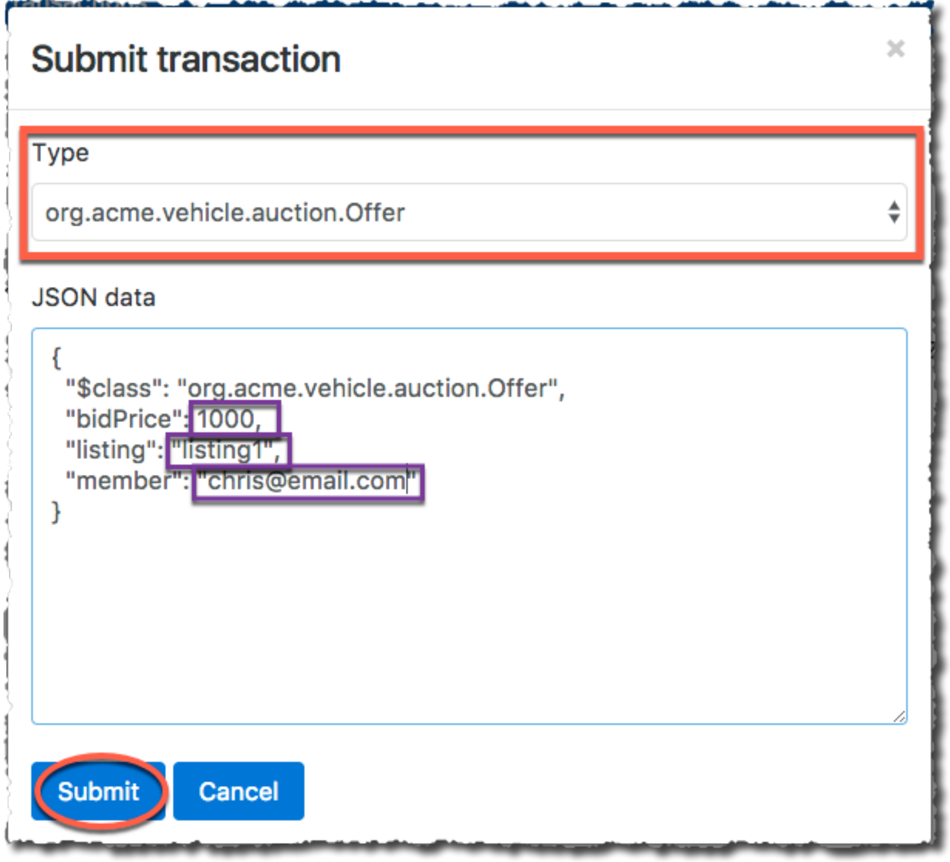
1.6. Submit a transaction to bid on the vehicle

__27. Click on the Transactions view and click the '+' to add a transaction



The transaction being added will update the value of an asset to a new value.

__28. Use the listing ID, member name and fill in a bid price then submit the bid transaction.



Type

org.acme.vehicle.auction.Offer

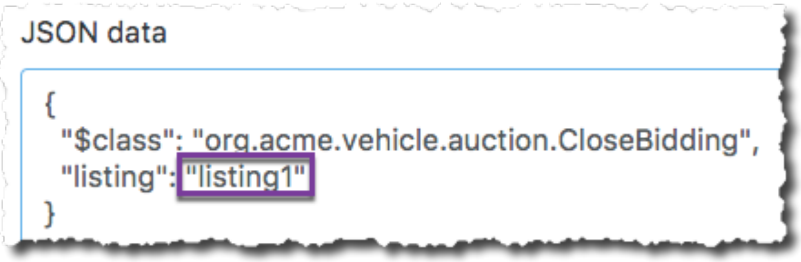
JSON data

```
{
  "$class": "org.acme.vehicle.auction.Offer",
  "bidPrice": 1000,
  "listing": "listing1",
  "member": "chris@email.com"
}
```

make sure the bidprice is over the reserve price set earlier

__29. View the transaction, note it has been given a transaction ID and timestamp by the system

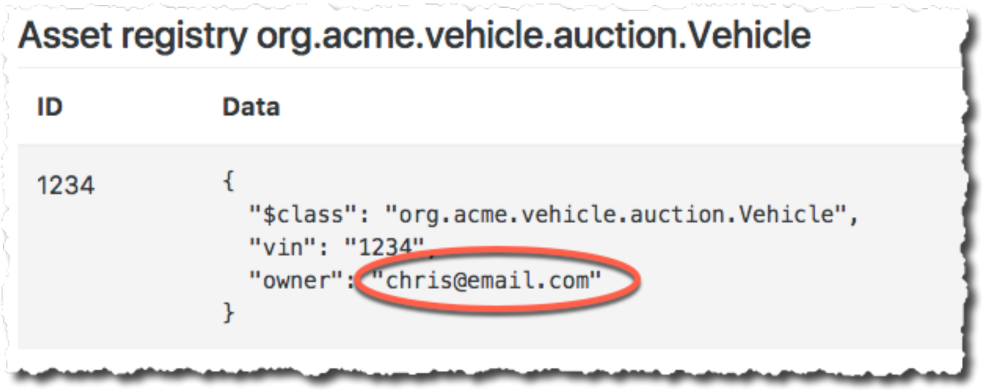
__30. Submit a new transaction, this time type: org.acme.vehicle.auction.ClosingBid for the same listing ID used previously



JSON data

```
{
  "$class": "org.acme.vehicle.auction.CloseBidding",
  "listing": "listing1"
}
```

__31. Go to the org.acme.vehicle.auction.Vehicle asset registry from the Assets view to see the vehicle owner has been updated



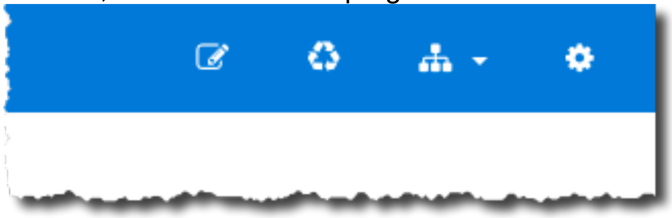
Asset registry org.acme.vehicle.auction.Vehicle

ID	Data
1234	`{ "$class": "org.acme.vehicle.auction.Vehicle", "vin": "1234", "owner": "chris@email.com" }`

```
{
  "$class": "org.acme.vehicle.auction.Vehicle",
  "vin": "1234",
  "owner": "chris@email.com"
}
```

1.7. View Other UI Functions

The Fabric Composer UI has other features we can explore further in future labs, but to get an understanding of what is available, let's look at the top right icons



✎ - This will load new samples to use. This is pulling samples from GitHub.

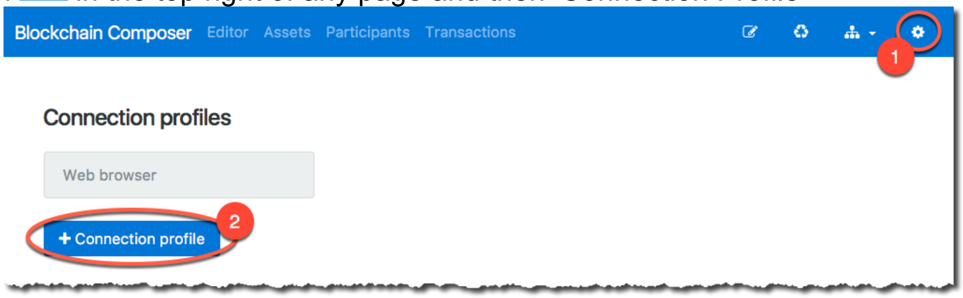
♻ - This will re-fresh the current sample back to its original state

🖧 - The default for this UI is to simulate all the network within the browser, this icon will allow you to connect to a netork that is defined within the connection profiles found within the settings section below.

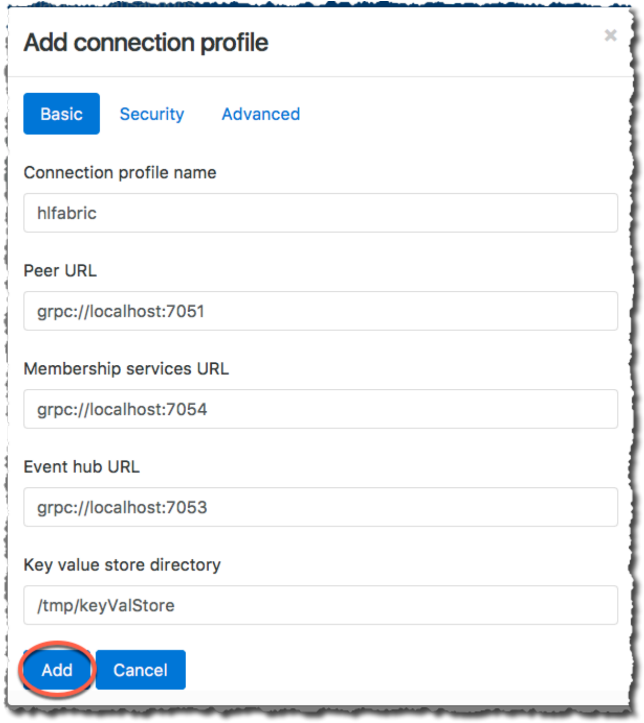
⚙ - The settings show the connection profiles to connect to specific networks

1.7.1. Understanding how could add to a live netwrok

__32. Click on ⚙ in the top right of any page and then ‘Connection Profile’



__33. This view shows how to set up a profile to connect to a live blockchain



Add connection profile

Connection profile name: hlfabric

Peer URL: grpc://localhost:7051

Membership services URL: grpc://localhost:7054

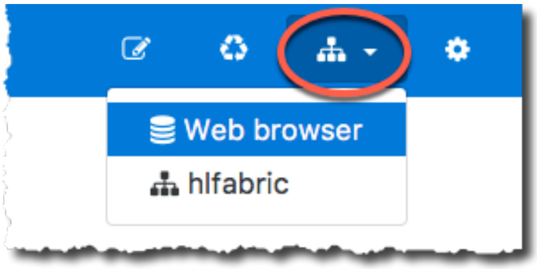
Event hub URL: grpc://localhost:7053

Key value store directory: /tmp/keyValStore

These are the default values if you had a hyperledger fabric running locally on your machine

__34. Click ‘Add’. Event though this setup will not connect to any system specifically, it will show the process

__35. From the top right, select the 🖧 icon to show all the connection profiles. This would be how to swap between the systems being worked on.

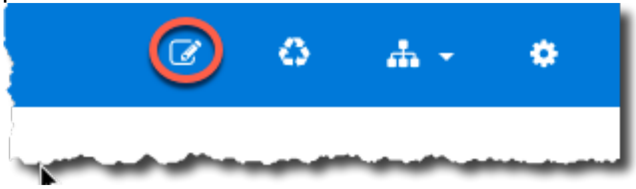


END OF LAB

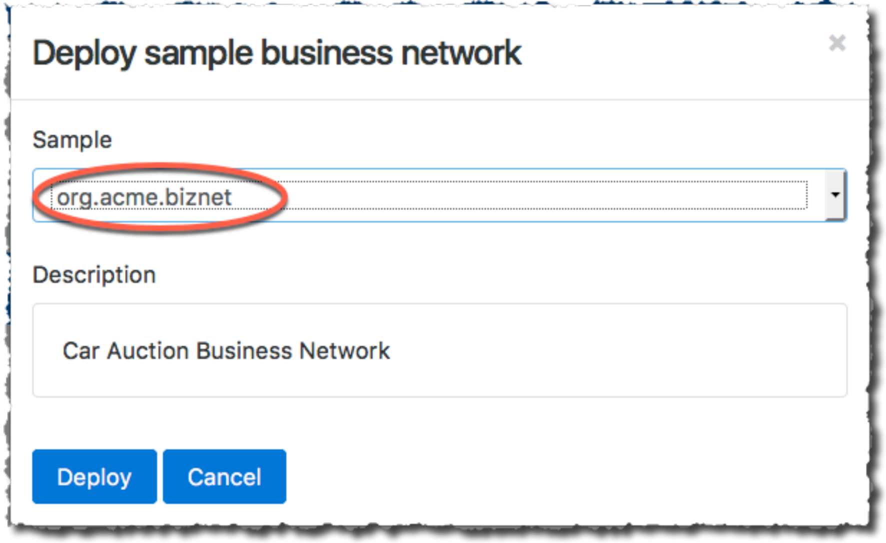
Appendix A. Reset the Demo

The Car Auction Demo can be re-set and as the demo is being run within the browser, it may need the browser to clear some information.

__1. Within the Demo, Open the ✎ to show demos that can be loaded



__2. Choose the **org.acme.biznet** from the dropdown as the Car Auction Network Demo



Deploy sample business network

Sample

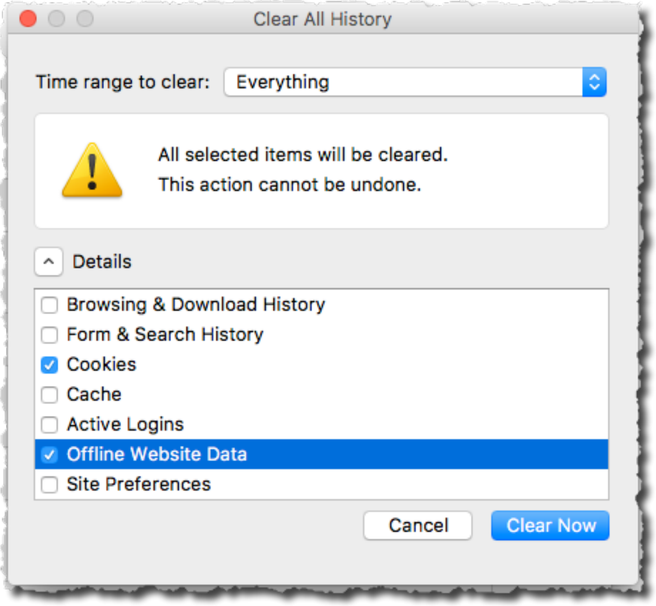
org.acme.biznet

Description

Car Auction Business Network

__3. Test if this is working

Occasionally the cache & Offline Website Data may need to be cleared to remove the old information



Clear All History

Time range to clear: Everything

All selected items will be cleared.
This action cannot be undone.

Details

- □ Browsing & Download History
- □ Form & Search History
- ☑ Cookies
- □ Cache
- □ Active Logins
- ☑ Offline Website Data
- □ Site Preferences

Appendix B. Notices

This information was developed for products and services offered in the U.S.A.
IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.
IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental. All references to fictitious companies or individuals are used for illustration purposes only.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Appendix D. Trademarks and copyrights

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	AIX	CICS	ClearCase	ClearQuest	Cloudscape
Cube Views	DB2	developerWorks	DRDA	IMS	IMS/ESA
Informix	Lotus	Lotus Workflow	MQSeries	OmniFind	
Rational	Redbooks	Red Brick	RequisitePro	System i	
System z	*Tivoli*	*WebSphere*	*Workplace*	*System p*	

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of The Minister for the Cabinet Office, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

NOTES

NOTES

© Copyright IBM Corporation 2014.

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. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

IBM, the IBM logo and ibm.com are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Please Recycle