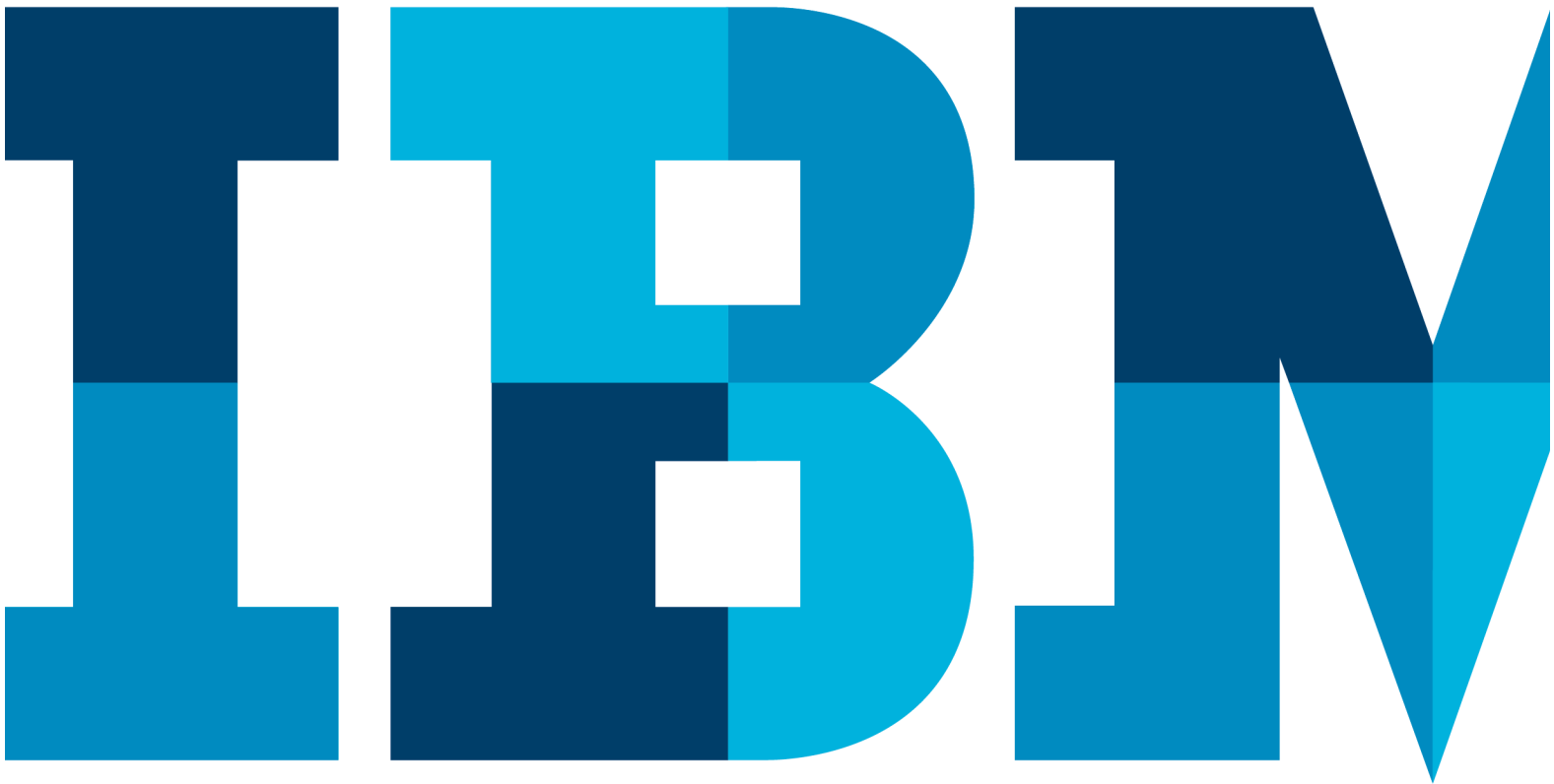


IBM Blockchain Hands-On The Composer Node.js SDK

Lab Two



Contents

SECTION 1.	CLI TOOL AND ARCHIVING	4
1.1.	STAND UP THE NETWORK	4
1.2.	CREATING ARCHIVES	4
1.3.	DEPLOYING ARCHIVES	4
1.4.	UPDATING ARCHIVES	5
SECTION 2.	MANIPULATING AND ADDING RESOURCES WITH THE SDK	6
2.1.	GETTING STARTED	6
2.2.	ADDING RESOURCES	7
2.3.	READING RESOURCES	9
2.4.	ADD MARBLES AND A NEW COLLECTOR	11
2.5.	ISSUING TRANSACTIONS	12
SECTION 3.	CLI TOOL AND IDENTITY MANAGEMENT	14
3.1.	ISSUING IDENTITIES	14
3.2.	USING IDENTITIES	14
3.3.	REVOKING IDENTITIES	16
NOTICES	17	
APPENDIX A.	TRADEMARKS AND COPYRIGHTS	19

Overview

The aim of this lab is to examine the node.js SDK for creating applications that interact with a Composer network. It will involve a brief overview of creating business networks with the command line tools and then mirror the latter part of Lab 1, creating and manipulating assets using the SDK. Identity management in Composer will also be explored in this lab and the effects of ACLs from the previous lab will be revisited.

Introduction

Pre-requisites:

- 4 cores
- 4GB RAM
- VMWare V10+
- The lab virtual machine

The virtual machine is based on Linux Ubuntu 16.04 and contains Hyperledger Fabric v1.0, Golang, Git, Visual Studio Code, Firefox, Hyperledger Composer v0.72.

A network needs to be visible to the virtual machine (even if the network is just to the host environment). If you do not see the up/down arrows in the status bar at the top of the screen, or if you receive errors about no network being available, please tell the lab leader. The virtual machine might need to be reconfigured in NAT mode.

There are no additional files or software that is proprietary to the lab in the virtual machine. This means that the lab may be run on a machine without the without a lab virtual machine if Hyperledger Fabric and the other pre-requisites have been installed.

It is recommended that students have previously completed the Blockchain Explained and Blockchain Explored labs.

Section 1. CLI Tool and Archiving

In this section we will look at how to create business network archive (.bna) files with the CLI tools. Please refer to the Setup Lab for reference on how these tools are set up on an Ubuntu machine.

1.1. Stand up the network

cd to ~/workspace/composer-getting-started and run start.sh

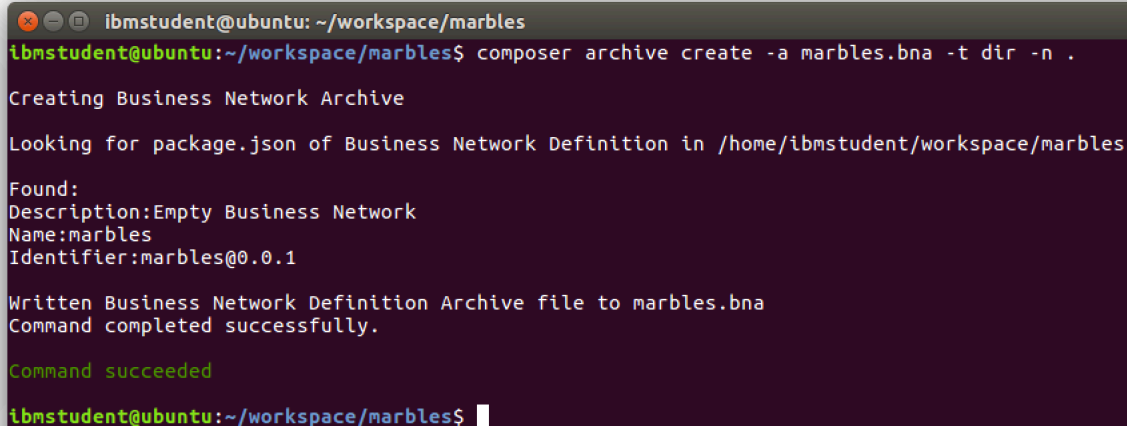
1.2. Creating Archives

Navigate to the directory that you extracted marbles into. Now it has been modified you need to package it back up into a business network archive or .bna file ready for re-upload back to the network.

To create an archive use the composer archive create command like so:

```
composer archive create -a marbles.bna -t dir -n .
```

This creates an archive called marbles.bna from the current directory and outputs it in the current directory. For more information on the command line tools issue composer --help or otherwise append the --help flag to any command for more information.



```
ibmstudent@ubuntu: ~/workspace/marbles
ibmstudent@ubuntu:~/workspace/marbles$ composer archive create -a marbles.bna -t dir -n .
Creating Business Network Archive
Looking for package.json of Business Network Definition in /home/ibmstudent/workspace/marbles
Found:
Description:Empty Business Network
Name:marbles
Identifier:marbles@0.0.1
Written Business Network Definition Archive file to marbles.bna
Command completed successfully.
Command succeeded
ibmstudent@ubuntu:~/workspace/marbles$
```

1.3. Deploying Archives

Now we have a business network archive we need to upload it to the network. To do so employs the composer network deploy command like so:

```
composer network deploy --archiveFile marbles.bna -p hlfv1 -i admin -s adminpw
```

This command takes an archive named `marbles.bna` and deploys it to the network specified in the `hlfv1` connection profile, authenticating the transaction with some credentials (in this case the username and password of the inbuilt admin account).

```
ibmstudent@ubuntu: ~/workspace/marbles
ibmstudent@ubuntu:~/workspace/marbles$ composer network deploy --archiveFile marbles.bna -p hlfv1 -i admin -s adminpw
Deploying business network from archive: marbles.bna
Business network definition:
  Identifier: marbles@0.0.1
  Description: Empty Business Network
✓ Deploying business network definition. This may take a minute...

Command succeeded
ibmstudent@ubuntu:~/workspace/marbles$
```

1.4. Updating Archives

If we make any edits to the business network it will have to be updated, to push the new version recompile the network definition into an archive file (as seen in step 1.2 above) and use the following command to update it:

```
composer network update --archiveFile marbles.bna -p hlfv1 -i admin -s adminpw
```

The syntax to this is identical to that of the deploy command.

```
ibmstudent@ubuntu: ~/workspace/marbles
ibmstudent@ubuntu:~/workspace/marbles$ composer network update --archiveFile marbles.bna -p hlfv1 -i admin -s adminpw
Deploying business network from archive: marbles.bna
Business network definition:
  Identifier: marbles@0.0.1
  Description: Empty Business Network
✓ Updating business network definition. This may take a few seconds...

Command succeeded
ibmstudent@ubuntu:~/workspace/marbles$
```

Section 2. Manipulating and Adding Resources with the SDK

In this section we will first look at how to connect to a running fabric instance with the composer node.js SDK and secondly follow this by looking at how to add and update resources. If you are stuck at any point please refer to the examples in ~/workspace/composer-samples/Lab 2/ for reference.

2.1. Getting Started

a. Examine the base code

Navigate to ~/workspace/composer-samples/Lab 2/marbles-client and open a file called skeleton.js with Visual Studio Code. it's contents should look like the following:

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

// Connect
bizNetConnection.connect("hlfv1", "marbles", "admin",
"adminpw").then(function(bizNetDef) {
    console.log("Connected to Network");

// Disconnect
}).then(function() {
    console.log("Done!");
    return bizNetConnection.disconnect();

// Exit wuth Success
}).then(function() {
    console.log("Disconnected from Network");
    process.exit(0);

// Catch any errors
}).catch(function (error) {
    console.log(error);
    process.exit(1);
});
```

This is the most rudimentary code to interact with a composer network running on fabric. It connects and then disconnects. Let's look at how.

b. Connecting

Connections to composer networks are established through a BusinessNetworkConnection object. This is a wrapper for the Hyperledger Fabric Client chain object that is customised to interact with Composer. The arguments in the connect function are as follows:

```
connect(<connection profile name>, <business network>, <username>, <secret>)
```

Connection profiles can be found in `$HOME/.composer-connection-profiles/` specify connection details for a fabric to connect to. The `hlfv1` profile was generated by `start.sh`.

c. Disconnecting

To disconnect from the composer network make a call to the `disconnect` method of the `BusinessNetworkConnection` object:

```
bizNetConnection.disconnect();
```

d. Promises

From inspecting the code above you will see that it is formed as a 'promise chain'. A promise is an object returned by asynchronous functions that upon their return triggers an attached callback (the `.then()` functions).

All of the Composer SDK's functions return promises, as such (as will be seen in this lab) code is often structured as a promise chain, with a series of `then()` methods resolving each section.

2.2. Adding Resources

a. Create a new file

`cd` back to `~/workspace` and create a new folder called `marbles-client` and `cd` into it.

Copy the contents of `skeleton.js` into a new file called `add-collector.js`.

b. Get the business network definition

Before and within the first promise callback add the following:

```
let factory;

bizNetConnection.connect("hlfv1", "marbles", "admin", "adminpw").then(function
(bizNetDef) {
    factory = bizNetDef.getFactory();
}) ...
```

The business network definition object is a representation of the business network and produces a factory object that is used to create new resources. Factories are objects that produced pre-formed but empty for us to use (i.e. objects with the current fields but no data). Therefore, we need to save the factory for later use.

c. Get the registry

To add a new resource to the network you need to add it to that resource's registry, as such we will need to fetch the registry object. Below the line of code capturing the factory add the following:

```
return bizNetConnection.getParticipantRegistry("org.acme.model.Collector");
```

This will return a promise so we need a new callback to capture the registry when it is returned. Add a new `then()` after the first and before the disconnection code to capture the registry:

```
...
return bizNetConnection.getParticipantRegistry("org.acme.model.Collector");
}).then(function(collectorRegistry) {
  // Add this
  // Disconnect
}).then(function() {
  console.log("Done");
  return bizNetConnection.disconnect();
})
```

d. Create a new resource

Within the body of this new callback, make a call to the factory:

```
let newCollector = factory.newResource("org.acme.model",
                                     "Collector",
                                     "tom@ibm.com");
```

As said before, factory objects create properly formed by empty objects for us to populate and use. The `newResource` function creates a resource based on the supplied namespace and resource name and assigns it the supplied id.

e. Populate it's attributes

Although we have our new resource, its fields are all empty. As such, we must populate it's attributes with the relevant details. Add the following below the declaration of `newCollector`:

```
newCollector.firstname = "Tom";
newCollector.surname = "Appleyard";

newCollector.address = factory.newConcept("org.acme.model", "Address");
newCollector.address.house = "IBM Bluemix Garage";
newCollector.address.street = "1 Fore Street";
newCollector.address.county = "London"
newCollector.address.postcode = "EC2Y 9DT";
newCollector.address.country = "United Kingdom";

newCollector.sex = "MALE";
```


Concepts are also created with the factory using the `newConcept` function. Much like with resources once a concept is created it's attributes must be populated. Factories are also used to create relationships between resources as we will see when we add a Marble.

f. Add it to the registry

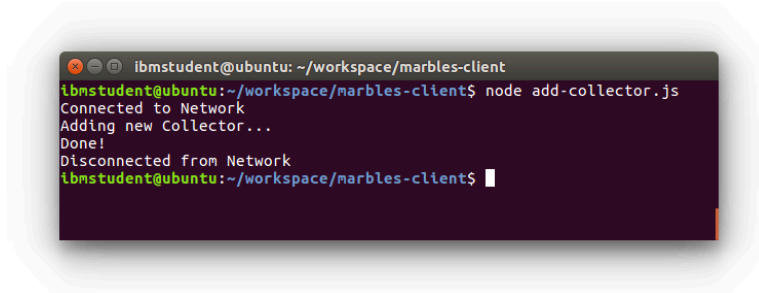
Finally the new resource must be added to the registry for it to exist on the system. Add the following below the code populating the resource:

```
return collectorRegistry.add(newCollector);
```

g. Test add-collector.js

To run the script, issue the following commands from within the `marbles-client` folder:

```
npm install composer-client
node add-collector.js
```



2.3. Reading Resources

h. Create a new file for reading

Copy the contents of `skeleton.js` into a new file called `read-resources.js`.

i. Get the resources

Resources are stored in registries in composer. Much like with adding resources, to read information we will first need the registry in which they are stored. Add the following in the body of the first promise callback:

```
bizNetConnection.connect("hlfv1", "marbles", "admin", "adminpw").then((bizNetDef) =>
{
    return bizNetConnection.getParticipantRegistry("org.acme.model.Collector");
}).then(function(collectorRegistry) {
    return collectorRegistry.getAll();
}).then(function(collectors) {
}).then(function() {
    return bizNetConnection.disconnect();
})
```

First a call is made to the `BusinessNetworkConnection` object to fetch the registry as was the case with the adding resource code. The registry is then asked to package all the assets it is storing into an array with `getAll()`. The promise chain is continued twice here – first to capture the registry and second to capture the array.

j. Print the resources

The `collectors` argument is an array of objects representing the resources in the registry. Their attributes are accessed in the same manner that any JavaScript object is accessed using dot notation i.e. `<object>.<attribute>`.

Add the following in the body of the `collectors` promise callback:

```
}).then(function(collectors) {
  let table = new Table({
    head: ['email', 'First Name', 'Last Name', 'Sex', 'Postcode']
  });

  collectors.forEach(function(collector) {
    let row = [];
    row.push(collector.email);
    row.push(collector.firstname);
    row.push(collector.surname);
    row.push(collector.sex);
    row.push(collector.address.postcode);
    table.push(row);
  });

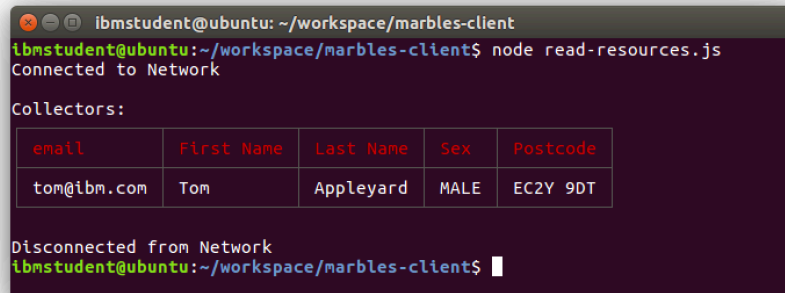
  console.log("Collectors:");
  console.log(table.toString());
  console.log("");
})
```

In addition to the above, below the `require` statement importing `BusinessNetworkConnection` at the top of the file add `const Table = require('cli-table');` this is a library for outputting formatted ascii tables.

k. Test read-resources.js

To test the read script run the following command:

```
npm install cli-table
node read-resources.js
```



```
ibmstudent@ubuntu: ~/workspace/marbles-client
ibmstudent@ubuntu:~/workspace/marbles-client$ node read-resources.js
Connected to Network

Collectors:


| email       | First Name | Last Name | Sex  | Postcode |
|-------------|------------|-----------|------|----------|
| tom@ibm.com | Tom        | Appleyard | MALE | EC2Y 9DT |


Disconnected from Network
ibmstudent@ubuntu:~/workspace/marbles-client$
```

2.4. Add Marbles and a new Collector

a. Marbles

Based on the instructions in 2.2 and 2.3, write an add-marble.js script. The only difference between the two processes is the following:

`bizNetConnection.getParticipantRegistry` is `bizNetConnection.getAssetRegistry`

As the owner attribute is a relationship you will need to employ the factory to create a relationship:

```
newMarble.owner = factory.newRelationship("org.acme.model",
                                         "Collector",
                                         "tom@ibm.com")
```

You should set the ID of the marble you will be inputting to MARBLE001. Once you have created add-marble.js, expand read-resources.js so that it reads marbles too.

b. Collectors

When you have finished Marbles, comment out the code in add-collector.js that adds tom@ibm.com and replace it with code that creates a collector with the following attributes:

```
email = "mgk@ibm.com"
firstname = "Matthew"
surname = "Golby-Kirk"
address.house = "IBM Hursley"
address.street = "Hursley Park"
```

```
address.county = "Hampshire"  
address.postcode = "SO21 2JN"  
address.country = "United Kingdom"  
sex = "MALE"
```

Run the updated code with `node add-collector.js`

2.5. Issuing Transactions

a. Create a new file

Copy the contents of `skeleton.js` into a new file called `transfer-marble.js`.

b. Create a new transaction

Add the following to the first promise callback:

```
bizNetConnection.connect("hlfv1", "marbles", "admin",  
"adminpw").then(function(bizNetDef) {  
  console.log("Connected to Network");  
  bizNetDefinition = bizNetDef;  
  factory = bizNetDef.getFactory();  
  
  // Create & Submit a new transaction  
  console.log("Submitting transaction...");  
  
  let newTransaction = factory.newTransaction("org.acme.model", "ChangeOwner");  
  newTransaction.marble = factory.newRelationship("org.acme.model", "Marble",  
                                                  "MARBLE001");  
  newTransaction.newOwner = factory.newRelationship("org.acme.model", "Collector",  
                                                    "mgk@ibm.com");  
  
  return bizNetConnection.submitTransaction(newTransaction);  
})
```

As you can see, transactions are created in an almost identical manner to other resources and are submitted by calling on the `BusinessNetworkConnection` to submit them.

c. Test

To test the transaction code, run the following:

```
node transfer-marble.js
```

```
ibmstudent@ubuntu: ~/workspace/marbles-client
ibmstudent@ubuntu:~/workspace/marbles-client$ node transfer-marble.js
Connected to Network
Submitting transaction...
Done!
Disconnected from Network
ibmstudent@ubuntu:~/workspace/marbles-client$
```

When this is complete, run `read-resources.js` again, the marble should have changed owners:

```
ibmstudent@ubuntu: ~/workspace/marbles-client
ibmstudent@ubuntu:~/workspace/marbles-client$ node read-resources.js
Connected to Network
Collectors:


| email        | First Name | Last Name  | Sex  | Postcode |
|--------------|------------|------------|------|----------|
| matt@ibm.com | Matt       | Golby-Kirk | MALE | S021 2JN |
| ton@ibm.com  | Ton        | Appleyard  | MALE | EC2Y 9DT |


Marbles:


| ID        | Owner        | Colour | Diameter |
|-----------|--------------|--------|----------|
| MARBLE001 | matt@ibm.com | red    | 14       |


Disconnected from Network
ibmstudent@ubuntu:~/workspace/marbles-client$
```

Section 3. CLI Tool and Identity Management

3.1. Issuing Identities

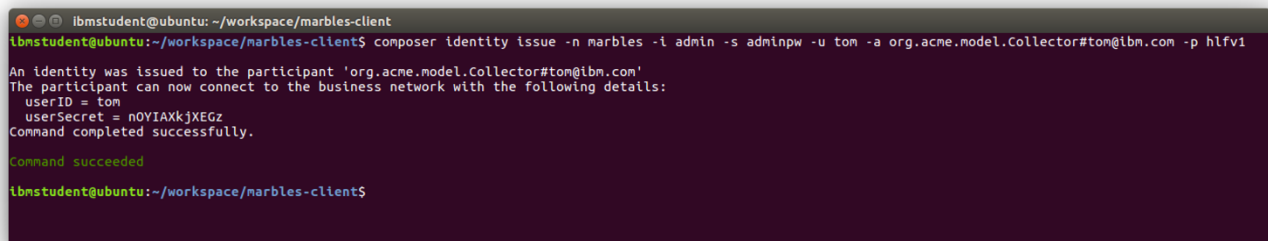
To issue an identity open a terminal and run the following command:

```
composer identity issue -n marbles -i admin -s adminpw -u tom -a
org.acme.model.Collector#tom@ibm.com -p hlfv1
```

The grammar is as follows:

```
composer identity issue -n <business network name> -i <username> -s <secret> -u
<username to be issued> -a <participant to be attached to> -p <connection profile>
```

You will be provided with a userID and a secret with which to verify your identity.



```
ibmstudent@ubuntu: ~/workspace/marbles-client
ibmstudent@ubuntu:~/workspace/marbles-client$ composer identity issue -n marbles -i admin -s adminpw -u tom -a org.acme.model.Collector#tom@ibm.com -p hlfv1
An identity was issued to the participant 'org.acme.model.Collector#tom@ibm.com'
The participant can now connect to the business network with the following details:
  userID = tom
  userSecret = nOYIAXkjXEGz
Command completed successfully.
Command succeeded
ibmstudent@ubuntu:~/workspace/marbles-client$
```

Issue another ID for mgk@ibm.com. Record these as they will only be presented to you once.

3.2. Using Identities

The username + secret combination presented can be used in any of the situations where composer has requested a username + secret. The most obvious example of this is in conjunction with the scripts written earlier.

Open up transfer-marble.js and replace the appropriate fields:

```
//bizNetConnection.connect("hlfv1", "marbles", "admin", "adminpw")
//bizNetConnection.connect("hlfv1", "marbles", "mgk", "uSUeHZQTjzly")
bizNetConnection.connect("hlfv1", "marbles", "tom", "nOYIAXkjXEGz")
```

Recall the ACL rule that restricts updating assets to their owners:

```
rule OnlyOwnerCanEdit {
  description: "Only an owner can edit a marble"
  participant(p): "org.acme.model.Collector"
  operation: UPDATE
  resource(r): "org.acme.model.Marble"
  condition: (r.owner.getIdentifier() == p.getIdentifier())
  action: ALLOW
}
```

Now run `transfer-marble.js` and see the different response:

State:

```
ibmstudent@ubuntu: ~/workspace/marbles-client
ibmstudent@ubuntu:~/workspace/marbles-client$ node read-resources.js
Connected to Network

Collectors:


| email        | First Name | Last Name  | Sex  | Postcode |
|--------------|------------|------------|------|----------|
| matt@ibm.com | Matt       | Golby-Kirk | MALE | S021 2JN |
| tom@ibm.com  | Tom        | Appleyard  | MALE | EC2Y 9DT |



Marbles:


| ID        | Owner       | Colour | Diameter |
|-----------|-------------|--------|----------|
| MARBLE001 | tom@ibm.com | red    | 14       |



Disconnected from Network
ibmstudent@ubuntu:~/workspace/marbles-client$
```

As MGK (not the owner):

```
ibmstudent@ubuntu: ~/workspace/marbles-client
ibmstudent@ubuntu:~/workspace/marbles-client$ node transfer-marble.js
Connected to Network
Submitting transaction...
{ Error: Error: Participant 'org.acme.model.Collector#matt@ibm.com' does not have 'UPDATE' access to resource 'org.acme.model.Marble#MARBLE001'
  at /home/ibmstudent/workspace/marbles-client/node_modules/fabric-client/node_modules/grpc/src/node/src/client.js:434:17 code: 2, metadata:
  Metadata { _internal_repr: {} } }
ibmstudent@ubuntu:~/workspace/marbles-client$
```

As Tom (the owner):

```
ibmstudent@ubuntu: ~/workspace/marbles-client
ibmstudent@ubuntu:~/workspace/marbles-client$ node transfer-marble.js
Connected to Network
Submitting transaction...
Done!
Disconnected from Network
ibmstudent@ubuntu:~/workspace/marbles-client$
```

As Tom (no longer the owner):

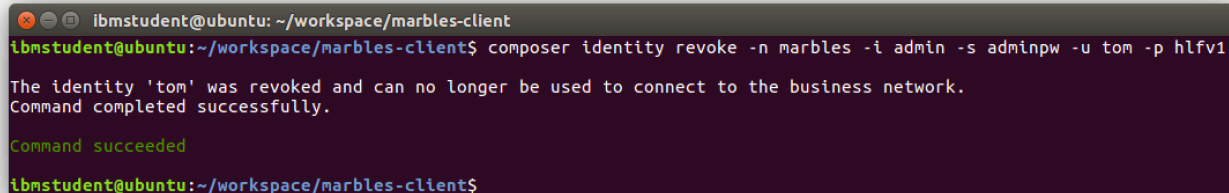
```
ibmstudent@ubuntu: ~/workspace/marbles-client
ibmstudent@ubuntu:~/workspace/marbles-client$ node transfer-marble.js
Connected to Network
Submitting transaction...
{ Error: Error: Participant 'org.acme.model.Collector#tom@ibm.com' does not have 'UPDATE' access to resource 'org.acme.model.Marble#MARBLE001'
  at /home/ibmstudent/workspace/marbles-client/node_modules/fabric-client/node_modules/grpc/src/node/src/client.js:434:17 code: 2, metadata:
  Metadata { _internal_repr: {} } }
ibmstudent@ubuntu:~/workspace/marbles-client$
```

3.3. Revoking Identities

Identities can of course be revoked as well as issued. To revoke an identity issue the following:

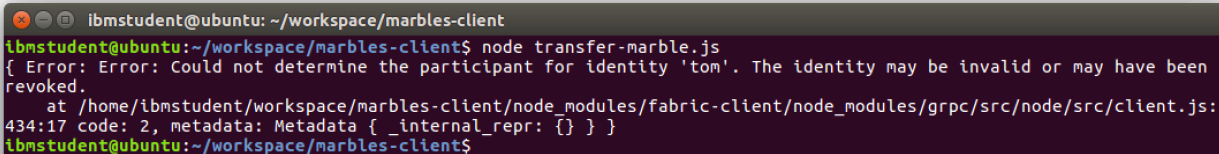
```
composer identity revoke -n marbles -i admin -s adminpw -u tom -p hlfv1
```

Flags denote the same as the previous command.



```
ibmstudent@ubuntu: ~/workspace/marbles-client
ibmstudent@ubuntu:~/workspace/marbles-client$ composer identity revoke -n marbles -i admin -s adminpw -u tom -p hlfv1
The identity 'tom' was revoked and can no longer be used to connect to the business network.
Command completed successfully.

Command succeeded
ibmstudent@ubuntu:~/workspace/marbles-client$
```



```
ibmstudent@ubuntu: ~/workspace/marbles-client
ibmstudent@ubuntu:~/workspace/marbles-client$ node transfer-marble.js
{ Error: Error: Could not determine the participant for identity 'tom'. The identity may be invalid or may have been
revoked.
  at /home/ibmstudent/workspace/marbles-client/node_modules/fabric-client/node_modules/grpc/src/node/src/client.js:
434:17 code: 2, metadata: Metadata { _internal_repr: {} } }
ibmstudent@ubuntu:~/workspace/marbles-client$
```

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 A. 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	
<i>System z</i>	<i>Tivoli</i>	<i>WebSphere</i>	<i>Workplace</i>	<i>System p</i>	

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

[illegible]

NOTES

[illegible]



© Copyright IBM Corporation 2016.

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
